MASTER EN SCIENCES INFORMATIQUES

Projet d'enseignement assisté par ordinateur : les systèmes d'équations linéaires

Jenné, Yves

Award date:
1984

Awarding institution:
Universite de Namur

PROJET D'ENSEIGNEMENT
ASSISTE PAR ORDINATEUR:

"LES SYSTEMES
D'EQUATIONS LINEAIRES."

Facultés Universitaires
Notre-Dame de la Paix
Institut d'Informatique

PROJET D'ENSEIGNEMENT ASSISTÉ PAR ORDINATEUR : LES SYSTÈMES D'ÉQUATIONS LINÉAIRES.

Année académique
1983-1984

Mémoire présenté
par Yves Jenné
en vue de l'obtention
du grade de licencié
et maître en informatique.

Promoteur : Cl. Cherton.

Je tiens tout d'abord à remercier Mademoiselle Wauthier et Monsieur Dupagne qui ont proposé ce mémoire et qui ont participé à sa réalisation.

Je veux également remercier Monsieur Cherton, promoteur de ce mémoire, pour m'avoir guidé dans ce travail.

Enfin, je remercie toutes les personnes qui m'ont apporté une aide quelconque. Je pense tout spécialement à Monsieur A. Berwaert pour le prêt fréquent de son matériel.

A mes parents,

A Jacqueline.

TABLE DES MATIERES.

INTRODUCTION.

Ce mémoire traite des systèmes d'équations linéaires. Son but est de servir d'aide à l'enseignement de cette matière dans les écoles du secondaire.
Qu'est ce qui a pu pousser des enseignants à nous demander un tel programme? Lors de nos premières discussions avec eux, ils nous ont appris que les élèves savaient en général résoudre les systèmes d'équations linéaires. Cependant les professeurs étaient peu convaincus du fait que les élèves se rendaient compte de ce qu'ils faisaient. Bien souvent les élèves avaient des difficultés pour interprèter leurs résultats. Ils ne voyaient pas de différence entre un système impossible et un système indéterminé, ni entre deux systèmes indéterminés d'ordres différents. Cela a conduit les enseignants à penser que les élèves, même s'ils appliquaient correctement des techniques de calcul, ne "voyaient" pas ce qu'ils faisaient.
Ces enseignants estimaient qu'un outil qui leur permettrait de visualiser rapidement et facilement des systèmes d'équations linéaires leur serait d'une grande utilité.
En clair, ce que les enseignants nous demandaient, c'était, pour un système quelconque :

1. De fournir un outil de résolution des systèmes, basé sur la méthode apprise par les élèves.
2. Si ce système admet une visualisation graphique assez simple (équations à 2 et 3 inconnues), de fournir également un support graphique. Celui-ci doit permettre de donner une autre vision des choses à l'élève. Dans le cas d'un sytème 3 x 3 par exemple, il faut
   - pouvoir visualiser chaque plan, ainsi que les intersections entre ceux-ci.
   - montrer, par un graphique clair, que si un tel système n'admet pas de solution, cela peut provenir de plusieurs raisons : 2 plans sont parallèles, les intersections entre les plans sont des droites parallèles ...
   - montrer à l'élève la différence entre un système indéterminé d'ordre 2 et d'ordre 1.
     Il est d'ordre 2 si les 3 plans sont confondus.
     Il est d'ordre 1 si tous les points solution se situent sur une même droite, c.a.d. si les 3 plans ont une droite commune.

De toute façon, nous espérons que dans les 2 cas, l'élève ne dira plus : " Si le système est indéterminé, la solution c'est n'importe quoi".

- faire mieux comprendre à l'élève ce qu'il fait quand il élimine une variable, car il pourra constater l'effet de cette élimination sur le graphique.

1. PREMIERE APPROCHE DU PROBLEME.

Dans ce chapitre nous allons dans une première partie préciser les objectifs de ce travail, ainsi que les moyens que nous avons retenus pour y arriver.
Dans la seconde partie, nous présentons le scénario d'une exécution du programme, ce qui permettra de voir toutes les possibilités que nous avons retenues.

1.1. DEFINITION DES BUTS ET DES OUTILS NECESSAIRES.

Le but de ce mémoire est bien sûr d'être un support informatique d'aide à l'enseignement des systèmes d'équations linéaires. Nous avons cru intéressant d'utiliser un micro-ordinateur à cet effet parce qu'il permet de donner aux élèves une autre représentation des systèmes, par la visualisation graphique des équations à 2 et 3 inconnues. Toutefois nous ne nous limiterons pas à 2 ou 3 inconnues, mais pour les ordres supérieurs nous ne présenterons pas de vue graphique.
A partir de là nous avons décomposé cet objectif principal en 3 parties:

1. Support graphique (pour 2 et 3 inconnues).
2. Résolution numérique des systèmes.
3. Dialogue avec l'utilisateur (pour lui proposer d'utiliser de plusieurs façons les 2 premières parties).

1.1.1. Support graphique.

Du point de vue pédagogique il s'agit sans doute de l'aspect le plus important de ce travail: les enseignants comptent beaucoup sur cette "représentation physique" des systèmes. (voir introduction).
Il s'agira de représenter à l'écran les systèmes 2x2 et 3x3 de manière claire et précise.
Pour les 2x2, il s'agira d'être capable de dessiner une droite d'équation $AX + BY = C$, à partir des trois coëfficients quelconques A, B, et C. Pour ce faire nous utiliserons un système d'axes orthogonaux, avec une même unité sur chaque axe, afin de donner une idée correcte de la pente.
Cette exigence conduira parfois à une certaine imprécision: si sur un axe nous devons représenter un point très éloigné de l'origine et si sur l'autre nous devons en représenter un très proche, cette deuxième distance sera parfois assimilée à 0. En conclusion, cette droite (qui est trés proche d'un des 2 axes) sera peut-être confondue avec cet axe. Pour le moment nous n'avons pas encore

envisagé de solution à ce problème, ni aux autres imprécisions graphiques que nous décrirons plus loin.

Il faudra également déterminer si deux droites d'équations données sont confondues, parallèles ou sécantes. Si elles sont parallèles nous nous assurons que les deux portions de droite que nous présentons à l'écran sont visibles.

Si elles sont sécantes, nous essaierons de visualiser aussi clairement que possible la position du point d'intersection. Il est certain que pour des cas extrêmes (par ex. : 2 droites sécantes dont les pentes sont très proches) la représentation du point d'intersection sera parfois imprécise; comme les droites sont graphiquement représentées comme des "escaliers", il se pourra que l'intersection vue à l'écran ne se limite pas à un seul point lumineux, mais à plusieurs.

Pour les 3x3, nous devons représenter convenablement tous les plans d'équation $AX + BY + CZ = D$, et cela quels que soient les coëfficients A, B, C et D. Nous déterminerons plus tard (voir 2.2) des conventions pour cela. Nous utiliserons un système de 3 axes, en conservant également la même échelle sur chaque axe. Dans certains cas, nous changerons alors de système de référence afin de permettre une (meilleure) vue graphique.

Nous représenterons aussi 2 et 3 plans simultanément. Selon la position de ces plans les uns par rapport aux autres (confondus, parallèles, sécants) nous déterminerons des portions de plan qui permettent de bien les situer dans l'espace (ainsi que leurs éventuelles intersections).

Il reste également un certain nombre de cas limites où la précision graphique risque d'être insuffisante.

Pour chaque segment de droite, nous retrouvons les mêmes problèmes que pour les systèmes 2x2 (droite très proche d'un axe, point d'intersection imprécis).

Lors de la superposition de 2 plans, il arrive aussi que le dessin ne soit pas très clair. Nous en donnons un exemple à la fin du point 2.2.2.

1.1.2. Résolution numérique des systèmes.

Nous désirons suivre la méthode utilisée par les élèves. Cela implique plusieurs choses :

- La méthode de résolution en elle-même; nous procédons par éliminations succesives de variables et par substitution des inconnues par leur valeur dès que celle-ci peut être déterminée.
- Les élèves travaillent peu avec des nombres réels. En général, ils utilisent des fractions pour effectuer leurs calculs. Nous allons donc faire de même. A cet effet nous devrons définir des opérations sur les fractions.

Nous constatons que ces exigences vont fortement déterminer la manière de réaliser cet algorithme. Par exemple le fait de travailler avec des fractions nous compliquera la tâche parce que les opérations ne sont pas prédéfinies, parce qu'il faudra vérifier si chaque fraction pourra être représentée par des entiers.
Toutefois cela nous a paru indispensable pour que l'élève puisse comparer sa démarche avec celle appliquée par le programme. Si à un moment donné l'élève traite une équation de la forme " 2/7 X+Y = - 13/9 " et que le programme lui propose " 0.285714 X + Y= -1.444444 " cela risque d'être trés inefficace.
De même, en utilisant une autre méthode de résolution nous ne savons plus lui "montrer" ce qu'il fait, ni suivre son raisonnement. Si par exemple nous utilisons le calcul matriciel, nous pourrions envoyer à l'élève un message du genre suivant :
" Nous avons calculé le déterminant de la matrice des coëfficients. Comme il est non-nul, ce système admet une solution unique qui est ... "
Cela n'apprendrait rien à l'élève si cette méthode ne lui a pas été enseignée. Il verra qu'il y a une solution unique si à la fin des éliminations et substitutions successives il aura pu attribuer une valeur bien précise à chaque inconnue.

1.1.3. Dialogue avec l'utilisateur.

Nous avons défini 2 outils qui nous semblaient nécessaires à la réalisation de notre objectif principal. Il faut encore définir la manière dont nous allons les utiliser.
Nous avons choisi de réaliser 2 grands types d'exercices: premièrement nous voulons résoudre des systèmes (ceux-ci sont introduits par l'utilisateur), deuxièmement nous voulons vérifier si les réponses qu'un élève propose sont correctes (pour des exercices qu'il a lui-même introduits ou que l'ordinateur a créés).
Dans le premier cas, l'optique est de montrer à l'utilisateur une démarche correcte de résolution, pour un exercice qu'il a lui même introduit. Il pourra également choisir la façon dont cette solution lui sera présentée, selon qu'il désire uniquement la solution finale ou qu'il veut suivre la démarche étape par étape (après chaque élimination ou substitution, nous lui présentons le système transformé). Dans chacun de ces deux cas il pourra aussi demander la représentation graphique (pour les systèmes 2x2 et 3x3, bien sûr).
Dans le second cas, il s'agit de permettre à l'élève de tester la validité de ses réponses. Dans ce cas et de toute façon, quelle que soit sa réponse il pourra demander à voir la démarche adoptée par l'ordinateur pour la com-

parer à la sienne. Il est certain que du point de vue pédagogique, cette approche de vérification aussi limitée est beaucoup trop minimaliste. Nous espérons que dans un développement ultérieur (les outils de base restant valables), la vérification sera plus poussée, par ex. situer l'endroit où une erreur a été commise... (voir chapitre sur les améliorations).

Nous prévoyons en plus une partie où l'élève sera totalement passif : il s'agit d'une démonstration des différents cas possibles pour les systèmes 2x2 et 3x3. Cela nous a semblé utile pour plusieurs raisons :

- Cela permettra à l'élève de sefamiliariser avec les représentations graphiques. (Nous en profiterons pour montrer quelques cas particuliers de droites et de plans).
- Cela sera utile lors d'une première explication de la signification graphique d'un système.
- Cela servira enfin à montrer à l'élève les différentes raisons pour lesquelles un système est impossible, indéterminé ou à solution unique.

Nous allons dans la seconde partie de ce chapitre, préciser exactement toutes les possibilités, en présentant les différentes grilles d'écran et leurs enchaînements possibles.

1.2. SCENARIO D'UNE EXECUTION DE PROGRAMME.

Nous commençons chaque exécution par un écran d'accueil de l'utilisateur et de présentation générale du logiciel. Il donne en outre deux remarques d'utilisation (voir Init 1).

Ce premier écran sera toujours présenté à la mise en route du programme. Ce n'est qu'à partir de l'écran suivant que l'utilisateur peut faire des choix. Il s'agit d'un menu principal (voir Inimenu 1...) C'est lui qui permet la première orientation. Il propose cinq choix :

1. L'utilisateur propose un exercice et l'ordinateur le résoud.
2. L'ordinateur vérifie la réponse à un problème.
3. L'ordinateur propose une démonstration pour les systèmes 3x3.
4. Idem pour les systèmes 2x2.
5. L'utilisateur quitte le programme.

En fonction du premier choix, la suite de la session aura l'allure suivante :

1. Nous rappelons son choix à l'utilisateur (voir Inic 1) et nous lui de-

mandons s'il est ou non familier avec cet outil (voir Inimenu 2). Si non, nous lui donnons les explications nécessaires pour qu'il puisse introduire son système (voir Init 2, Init 3, Init 4). Ces explications portent sur des conventions de forme des coëfficients, des variables et des équations.

Lorsqu'il a introduit son système, nous lui offrons la possibilité de le modifier . Lorsqu'il ne veut plus rien y changer, nous lui proposons 4 types de solution selon qu'il désire ou non voir les différentes étapes du calcul, avec ou sans représentation graphique (voir Inimenu 3). Lorsque la solution retenue lui a été présentée, nous lui laissons le choix entre 3 possibilités pour continuer la session (voir Inimenu 4):

- Il peut proposer un autre exercice à résoudre par l'ordinateur.
- Il peut repartir de l'exercice précédent et en modifier certaines équations.
- Il peut retourner au menu principal.

2. Nous rappelons à l'utilisateur qu'il a demandé que l'ordinateur vérifie ses réponses (voir Inic 2).

Il doit ensuite choisir de résoudre soit un système qu'il introduit lui-même, soit un système créé par l'ordinateur (voir Inimenu 5).

Selon ce choix, soit nous lui proposons d'introduire son système (voir point 1), soit nous appelons une routine de création d'exercices et nous affichons le système créé (voir aussi point 2.7 sur la création d'exercices).

Nous demandons ensuite si le système qu'il a résolu est impossible, indéterminé ou s'il admet une solution unique (voir Inimenu 6).

Dans le premier cas il n'y a qu'à regarder si le système est bien impossible. Dans le second nous nous limitons (comme expliqué plus haut) à vérifier l'ordre d'indétermination. Dans le troisième cas, nous demandons à l'utilisateur d'introduire les valeurs qu'il a trouvées et nous les comparons avec celles calculées par le micro-ordinateur.

Après vérification de la "validité" (dans les limites précisées ci-dessus) de la solution proposée, nous offrons plusieurs choix:

- Si la réponse est correcte, l'utilisateur peut voir le cheminement numérique nécessaire à l'élimination avec ou sans graphique. Il peut aussi continuer ce genre d'exercices ou revenir au menu principal (voir Inimenu 7).
- Si la réponse est incorrecte, l'utilisateur peut demander à voir une des quatres présentations de la solution, il peut aussi recom-

mencer cet exercice, en demander un autre ou retourner au menu principal (voir Inimenu 8).

A ces quatre choix, nous aimerions en voir ajouter un cinquième qui serait "explication de l'erreur".

Nous y reviendrons au chapitre qui traite des prolongements nécessaires à ce travail.

3. Nous faisons appel à la "démonstration" des systèmes 3x3. Celle-ci présente d'abord quelques plans pour familiariser les élèves avec la représentation à l'écran.

Nous montrons par des exemples dans quels cas un système peut être impossible, indéterminé ou à solution unique.

Toute cette démonstration se déroule sans participation active de l'utilisateur (sauf pour fixer la vitesse du défilement des écrans).

A la fin de la démonstration nous revenons au menu principal et nous attendons un nouveau choix.

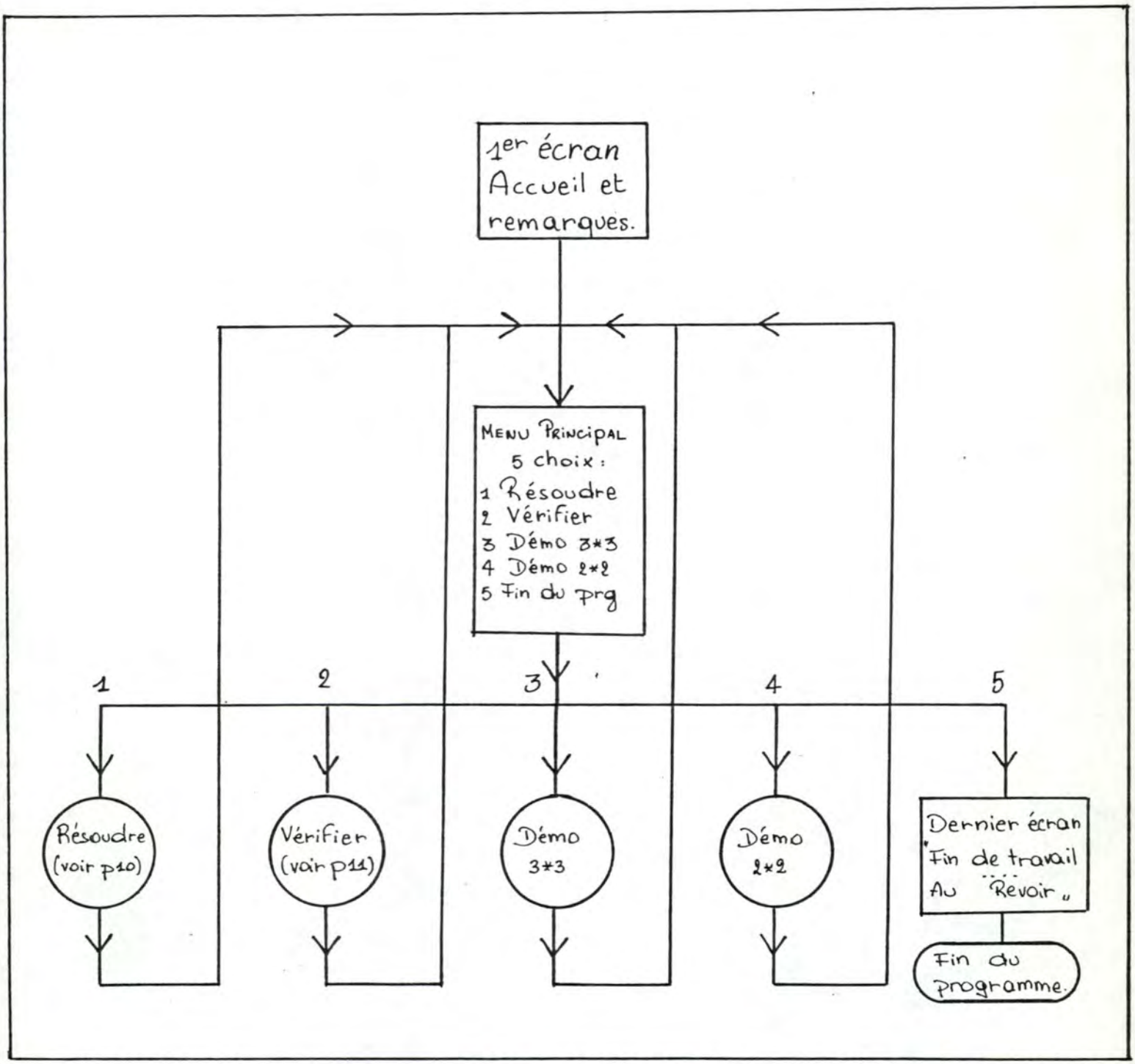
4. Nous appliquons le même principe pour les systèmes 2x2. Nous présentons d'abord quelques droites particulières et ensuite les différents cas de systèmes.

5. Nous affichons un écran qui signale la fin de la session (voir Init 5).

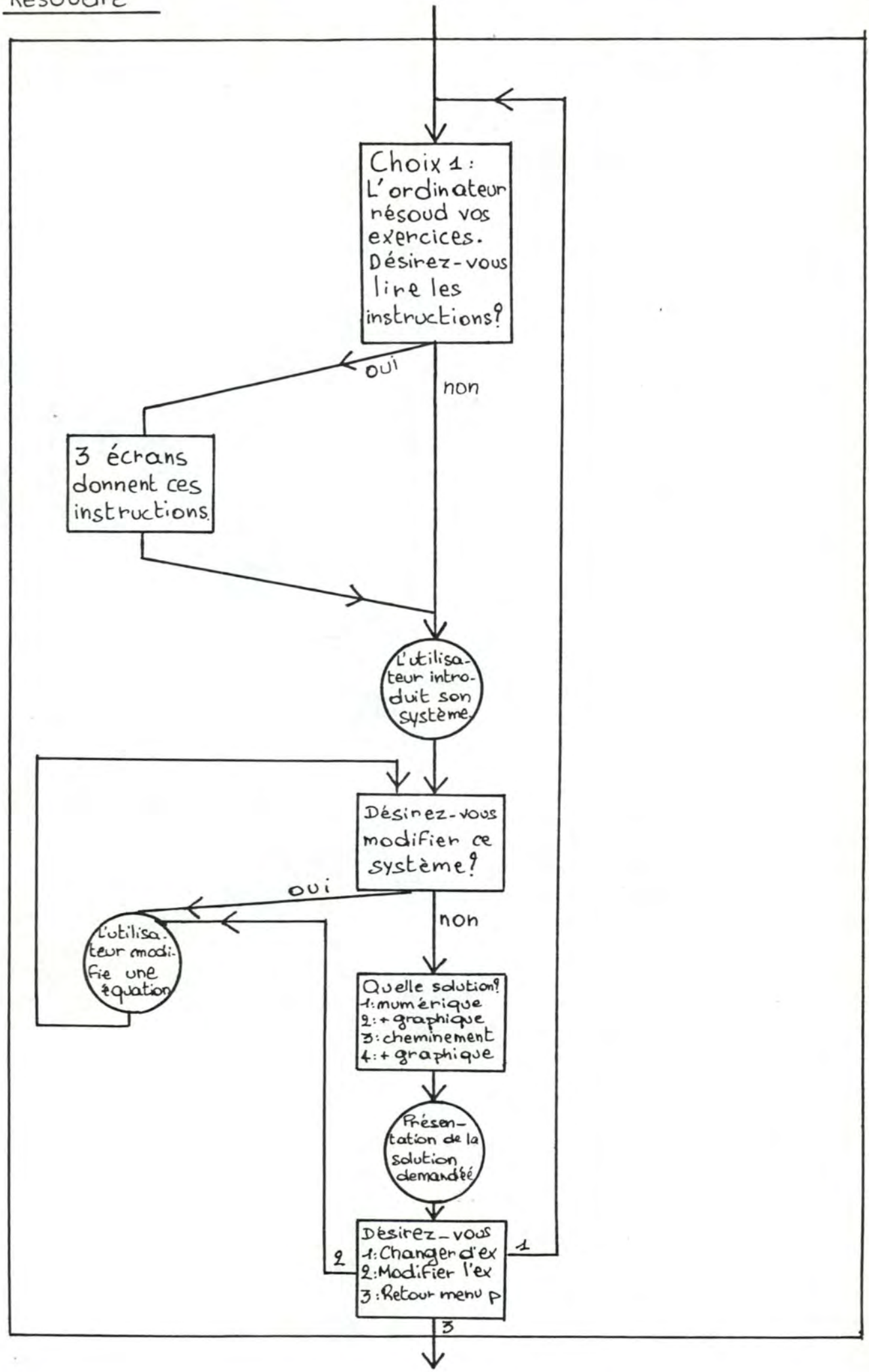
Remarquons qu'il s'agit de la seule façon "normale" d'arrêter le programme. Nous n'offrons jamais la possibilité de terminer la session ailleurs que dans le menu principal. Cela ne constitue pas une grosse contrainte pour l'utilisateur car il peut accéder au menu principal à partir de tous les autres. Par contre cela évitera peut-être à l'utilisateur maladroit de quitter le programme par distraction.

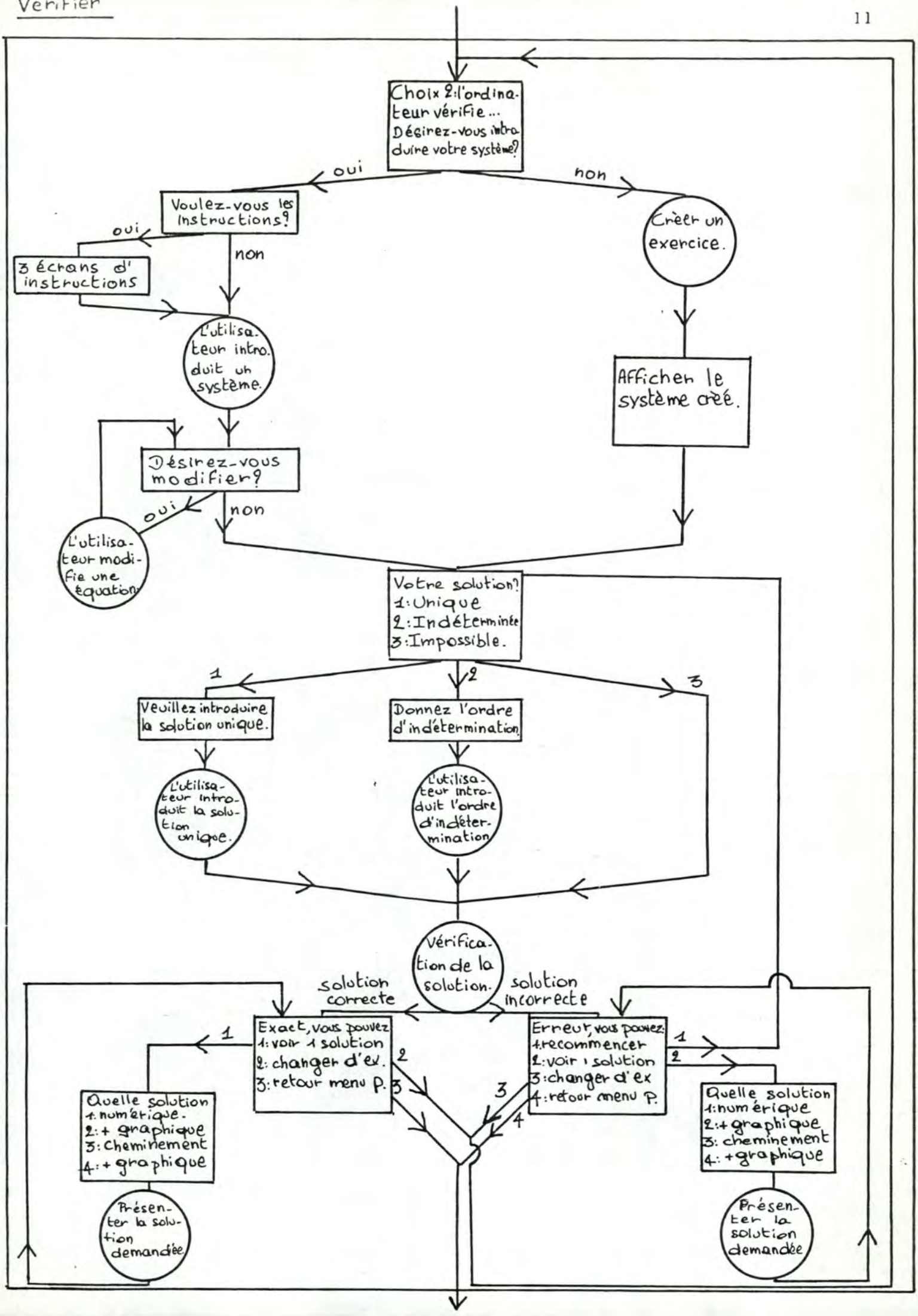
Sur les 3 diagrammes suivants nous allons schématiser les différents cheminements possibles:

- les rectangles représentent un écran présenté à l'utilisateur.
- les cercles représentent un appel à une procédure qui permet de réaliser l'action qui y est inscrite.
- les flèches représentent des relations de chronologie puisque le but de ces schémas est de montrer comment s'enchainent les écrans et les procédures selon les choix de l'utilisateur.


1er écran
Accueil et
remarques.
MENU PRINCIPAL
5 choix :
1 Résoudre
2 Vérifier
3 Démo 3*3
4 Démo 2*2
5 Fin du prg
1
2
3
4
5
Résoudre
(voir p10)
Vérifier
(voir p11)
Démo
3*3
Démo
2*2
Dernier écran
Fin de travail
Au Revoir "
Fin du
programme.

Résoudre


Choix 1 : L'ordinateur résoud vos exercices. Désirez-vous lire les instructions?
oui
non
3 écrans donnent ces instructions
L'utilisateur introduit son système
Désirez-vous modifier ce système?
oui
non
L'utilisateur modifie une équation
Quelle solution?
1: numérique
2: + graphique
3: cheminement
4: + graphique
Présentation de la solution demandée
Désirez-vous
1: Changer d'ex
2: Modifier l'ex
3: Retour menu p
1
2
3


Choix 2: l'ordinateur vérifie ... Désirez-vous introduire votre système?
oui
non
Voulez-vous les instructions?
oui
non
3 écrans d'instructions
L'utilisateur introduit un système.
Créer un exercice.
Affichen le système créé.
① désirez-vous modifier?
oui
non
L'utilisateur modifie une équation
Votre solution?
1: Unique
2: Indéterminée
3: Impossible.
1
2
3
Veuillez introduire la solution unique.
Donnez l'ordre d'indétermination
L'utilisateur introduit la solution unique.
L'utilisateur introduit l'ordre d'indétermination
Vérification de la solution.
solution correcte
solution incorrecte
Exact, vous pouvez
1: voir 1 solution
2: changer d'ex.
3: retour menu P.
Erreur, vous pouvez:
1: recommencer
2: voir 1 solution
3: changer d'ex
4: retour menu P.
Quelle solution
1: numérique.
2: + graphique
3: Cheminement
4: + graphique
Présenter la solution demandée
Quelle solution
1: numérique
2: + graphique
3: cheminement
4: + graphique
Présenter la solution demandée

Nous pouvons remarquer qu'à la fin de la partie "vérifier" lorsque nous avons présenté une des formes de solution, nous revenons au même menu. Nous procédons ainsi parceque nous ne voulons pas obliger l'élève à regarder une solution, ce qui serait le cas si nous utilisions la même structure que dans "résoudre".

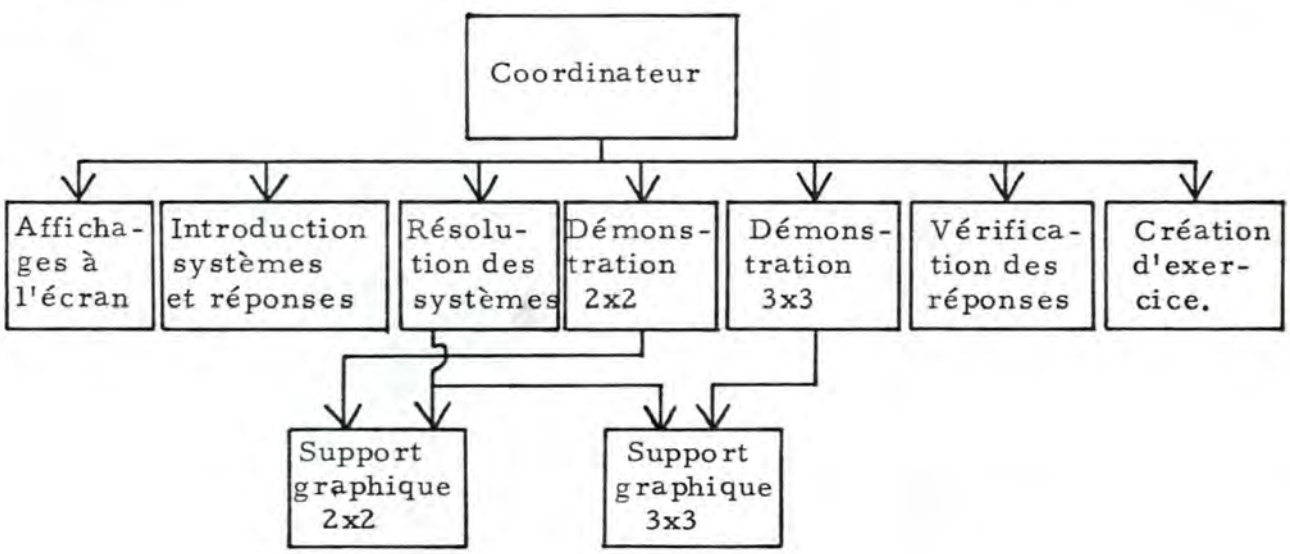
Ce n'est pas la seule solution envisageable. On pourrait par exemple "obliger" (c.a.d. ne pas lui offrir de choix) l'élève qui a commis une erreur de recommencer cet exercice. Ou bien lui montrer directement la solution.

2. DECOUPE GENERALE DU PROGRAMME.

Au chapitre précédent nous avons vu l'allure générale du programme que nous voulons réaliser. Rappelons que nous avons besoin d'un support graphique pour les systèmes 2x2 et 3x3. Nous devons également être capables de résoudre des systèmes, de vérifier des réponses et de créer des exercices. (Nous savons que nous serons soumis pour cela à quelques contraintes dues à des raisons pédagogiques). Il faut aussi permettre à l'utilisateur d'introduire des sytèmes et des réponses. Nous avons prévu ensuite une démonstration pour les systèmes 2x2 et 3x3.

Enfin il faudra guider l'utilisateur et lui proposer les différents choix qu'il peut effectuer à tout moment.

A partir de cela nous pouvons schématiser la structure du programme comme suit :



La relation définie par les flèches est une relation du type "appelle".

Dans ce chapitre nous allons spécifier exactement ce que doit réaliser chacun de ces modules.

2.1 LE SUPPORT GRAPHIQUE 2x2.

Le but de ce module est de représenter graphiquement un système de deux équations à 2 inconnues.

Il y a donc deux choses à réaliser :

- Représenter sur un graphique une équation à 2 inconnues, c'est à dire en général une droite.
- Représenter sur un même graphique deux de ces équations, pour montrer ce que sera la solution de ce système.

Considérons l'équation AX + BY = C. Pour la représenter à l'écran nous y dessinons un système d'axes de référence . Nous allons situer la droite par rapport à ces axes. Pour ce faire, nous étudierons la valeur des coëfficients A, B et C.

Il y a deux cas initiaux où cette équation ne peut être représentée par une droite :

- Si A, B et C sont tous les trois nuls l'équation devient 0X +0Y = 0. Cette expression est toujours vraie, pour toutes les valeurs de X et Y. Tous les points du plan sont donc solutions.
- Si A et B sont nuls et si C est non nul, l'équation est 0X + 0Y = C. ($C \neq 0$). Cette expression est toujours fausse; il n'existe aucune valeur de X et Y qui corresponde à cette expression.

Dans tous les autres cas, nous traçerons une portion de droite dans le système d'axes de telle manière que celle-ci soit visible sur l'écran. Pour cela, nous déterminons une unité et deux points de cette droite qui seront les extrémités du segment qui la représentera.

Cette unité dépendra du type de droite à représenter. Si celle-ci n'a pas d'intersection avec les axes elle pourra être arbitraire. Sinon nous la choisirons de manière telle que le (ou les deux) point(s) d'intersection avec les axes soi(en)t visible(s) sur l'écran. Selon que les coëfficients soient ou non nuls, la droite peut être parallèle ou confondue avec un des axes, couper les axes uniquement au point origine (0, 0) ou encore avoir deux points d'intersection distincts avec ceux-ci. Dans les différents cas il faut trouver les deux "points frontières" de la droite avec l'écran.

Pour représenter un système 2x2, sauf cas particuliers (A et B nuls), nous représenterons 2 portions de droite sur un même graphique.

Selon que les pentes des 2 droites sont égales ou non, nous aurons affaire à 2 droites parallèles (ou confondues) ou à 2 droites sécantes.

Dans le premier cas nous représentons les 2 droites dans les mêmes axes en utilisant bien sûr la même unité pour chacune.

Dans le second cas, nous déterminons également où se situe le point d'intersection et éventuellement nous modifions l'unité pour que ce point soit visible sur l'écran. Nous rappelons ici les réserves que nous avons émises quant à la précision de ce point sur le graphique. (voir point 1.1.1).

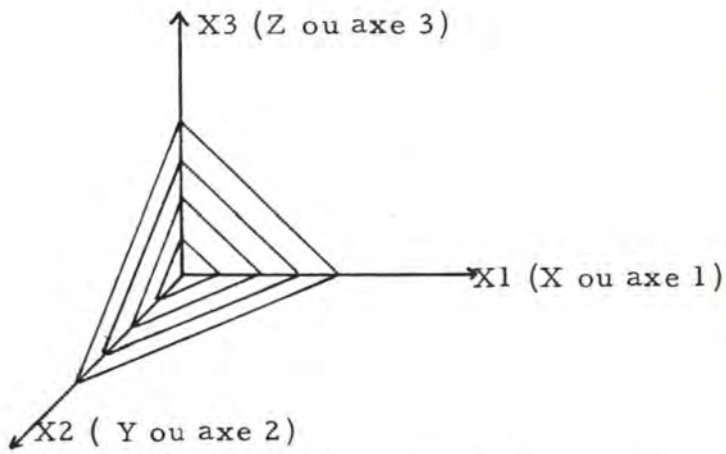
2.2 LE SUPPORT GRAPHIQUE 3x3.

Le but du module est de représenter graphiquement un système de 3 équations à 3 inconnues.
Nous avons décidé de procéder par étapes : nous présentons d'abord chaque équation séparément, puis les équations deux par deux et enfin le système complet.
Nous devons donc être capable de :

- Représenter graphiquement les équations du type AX + BY + CZ = D.
- Représenter simultanément 2 de ces équations avec leur éventuelle intersection.
- Représenter le système complet.

2.2.1. Représentation graphique d'une équation du type AX+BY+CZ=D.

La première chose à faire est de chercher un système assez habituel et utilisé dans le secondaire.



Le plan compris entre les axes X2 et X3 représente le plan debout.
Le plan compris entre les axes X1 et X3 représente le plan de face.
Le plan compris entre les axes X1 et X2 représente le plan horizontal.

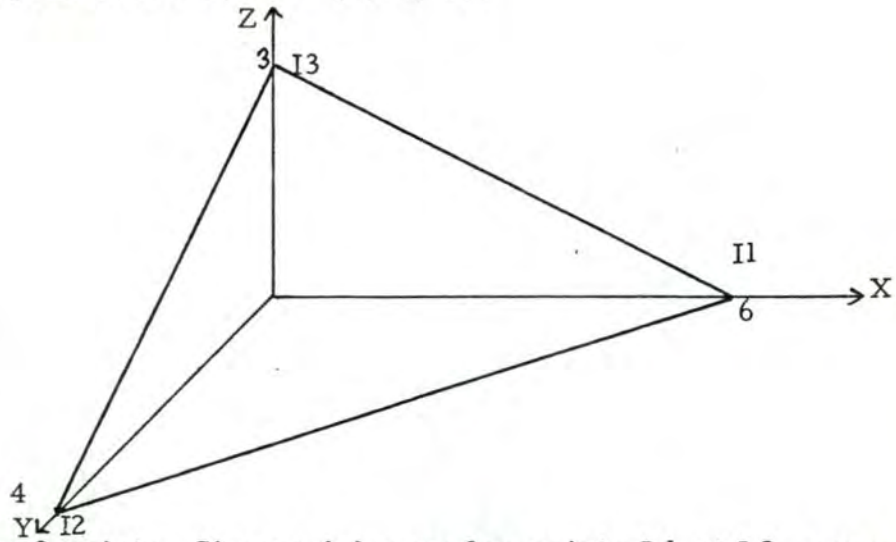
Nous utilisons la perspective cavalière pour représenter nos plans et nous choisirons une seule unité pour tous les axes, de manière à conserver toujours la proportion entre les différentes distances à y représenter.
Nous verrons plus loin que nous modifierons parfois la position de l'axe X2, selon les plans à représenter.
Nous allons prendre des conventions pour représenter les équations à 3 inconnues par une portion de plan (sauf les équations du type 0 = 0 ou 0 = D); notre idée est de partir de points faciles à visualiser (les intersections du plan avec les 3 axes, si elles existent).
Considérons l'exemple suivant: 2X + 3Y + 4Z = 12.

Nous obtenons les 3 points suivants :

I 1 = intersection du plan avec l'axe X = (6, 0, 0).

I 2 = intersection du plan avec l'axe Y = (0, 4, 0).

I 3 = intersection du plan avec l'axe Z = (0, 0, 3).



La plan passe par les 3 points. Si nous joignons les points I 1 et I 2, nous obtenons une partie de la droite d'intersection entre le plan à représenter et le plan horizontal.

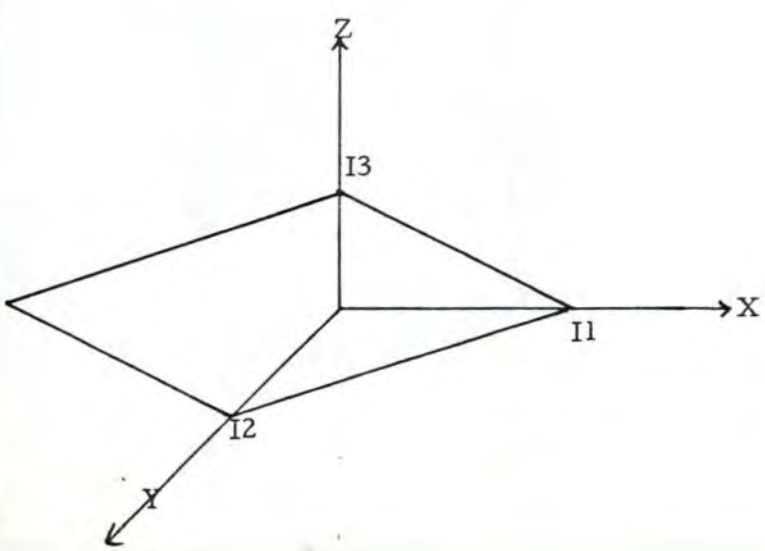
De même si nous joignons les points I 1 et I 3 et I 2 et I 3 nous obtenons respectivement des parties des droites d'intersection du plan à représenter avec les plans de face et debout.

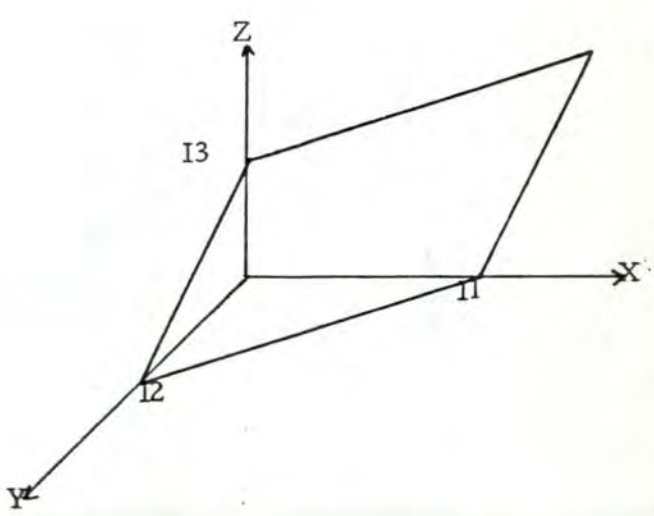
Nous avons ainsi défini un triangle qui permet de situer assez bien le plan, parce que à la fois les 3 points de référence sont faciles à situer, mais également les 3 droites qui les relient.

Cependant cette représentation ne nous a pas semblé appropriée, parce qu'en général, les professeurs introduisent la notion de plan en prenant comme exemple le mur, le tableau ou une feuille de papier.

Nous essayons alors de représenter un plan par un quadrilatère, tout en gardant ces 3 points comme référence, ainsi que la partie de droite reliant I 1 à I 2.

Il reste donc deux possibilités.:



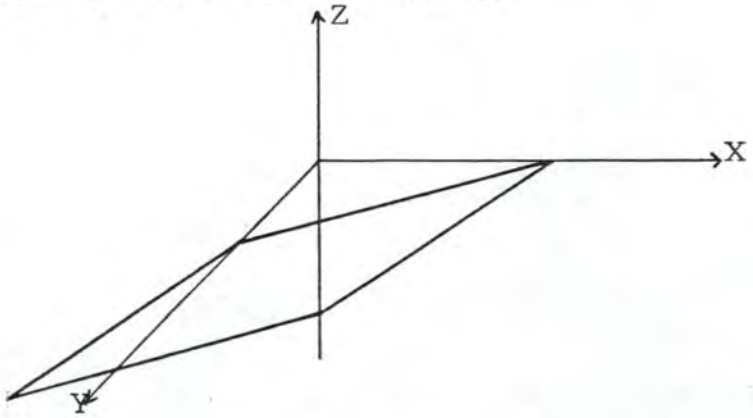


Après plusieurs essais graphiques, nous avons opté pour la première possibilité. Celle-ci nous a semblé plus suggestive au premier coup d'oeil. Nous pensons que ce schéma est assez "habituel" et que peu d'acclimation et de temps sont nécessaires pour pouvoir le "décoder".

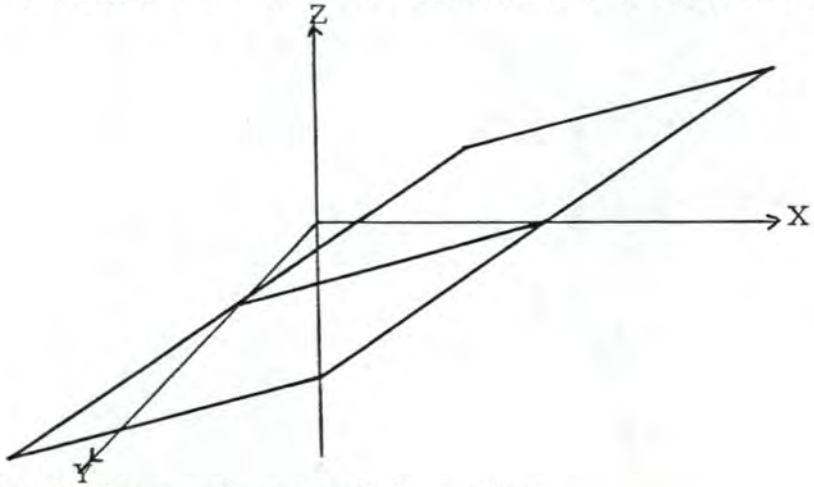
Nous retenons donc le principe de représentation suivant (si le plan a un point d'intersection avec chaque axe).

- Nous représentons une partie de la droite d'intersection avec le plan horizontal (I 1 - I 2).
- Idem avec le plan de face (I 1 - I 3).
- Nous traçons les parallèles à ces droites en partant de I 3 et I 2 et nous obtenons un parallélogramme.

Au cours de nos essais nous nous sommes aperçus que si les points I 1, I 2 ou I 3 étaient situés différemment sur les axes, il était dans certains cas possible d'améliorer la visualisation: voyons d'abord ce qui se passe si le point I 3 est situé sous le niveau de l'horizontale.

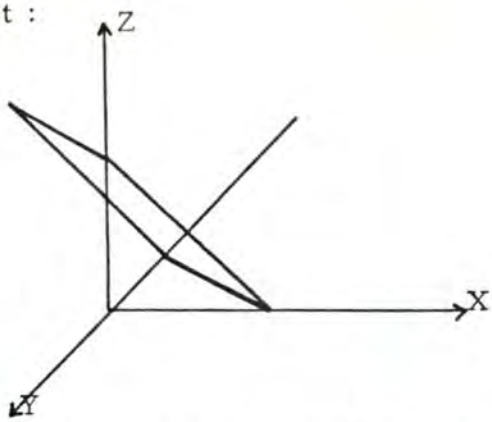


La vision de ce plan nous semble un peu moins bonne, sans doute parce que toute la portion de plan qui est dessinée est située sous le niveau du plan horizontal. Pour faciliter la visualisation, nous représentons également une partie du plan au-dessus de ce niveau horizontal. Nous obtenons :



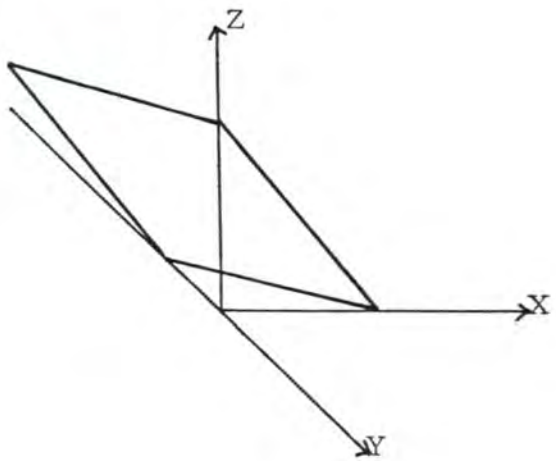
ce qui facilite la "compréhension" du schéma.

Voyons l'exemple suivant :



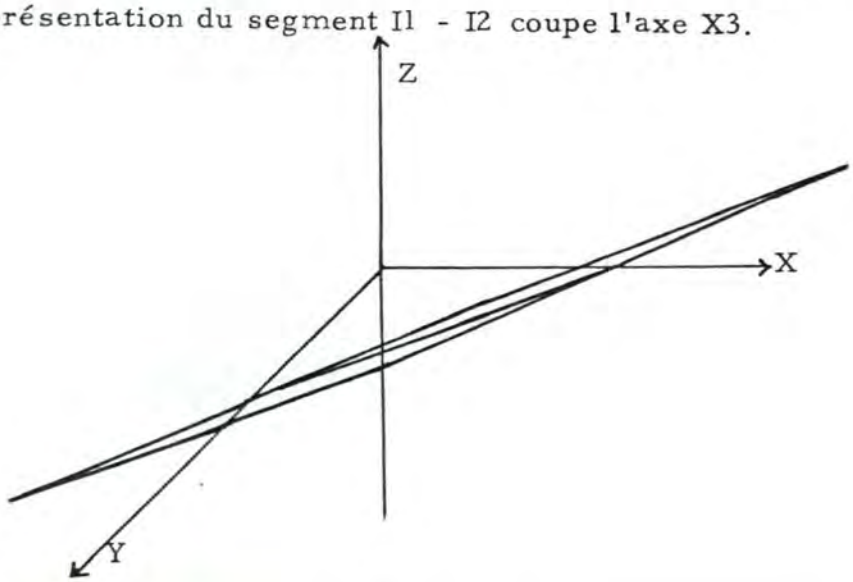
Nous pouvons améliorer également cette représentation. Si nous modifions le système d'axes de référence de telle manière que l'angle entre la "partie positive" de l'axe X et la "partie négative" de l'axe Y soit plus grand, la longueur de la portion de droite que nous représentons sera plus longue et la figure représentée moïns "raplatie".

Nous obtenons :

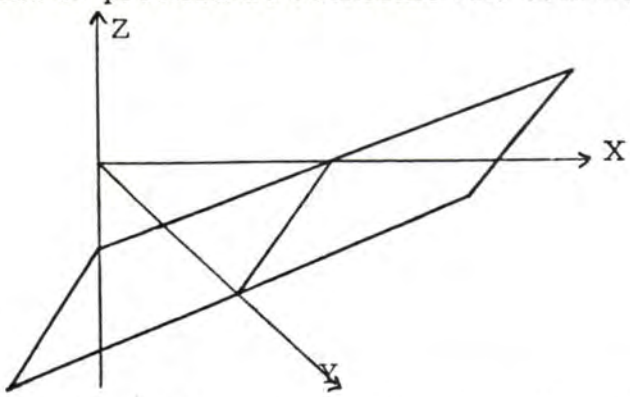


Ainsi si I 1 est situé dans la partie positive de l'axe 1 et que I 2 est situé dans la partie négative de l'axe 2 (ou inversément) nous changeons la position de l'axe 2 pour améliorer la visualisation.

Nous aurons également recours à ce deuxième système de référence pour représenter certains plans qui ne sont pas du tout "visibles" autrement. Par exemple, si le point I 3 se représente sur le schéma plus ou moins où la représentation du segment I1 - I2 coupe l'axe X3.

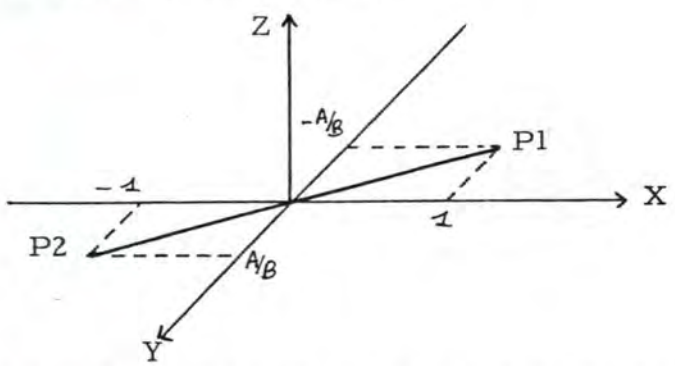


Par contre si nous changeons la position du deuxième axe comme tout à l'heure :



Nous avons choisi cette méthode afin de pouvoir conserver la même unité sur chaque axe. Nous n'avons pas voulu modifier l'unité sur l'un ou l'autre axe car alors la position du plan dans l'espace est faussée. A la limite, avec cette méthode, on pourrait représenter tous les plans d'un même type de la même façon, en changeant simplement les unités sur les axes.

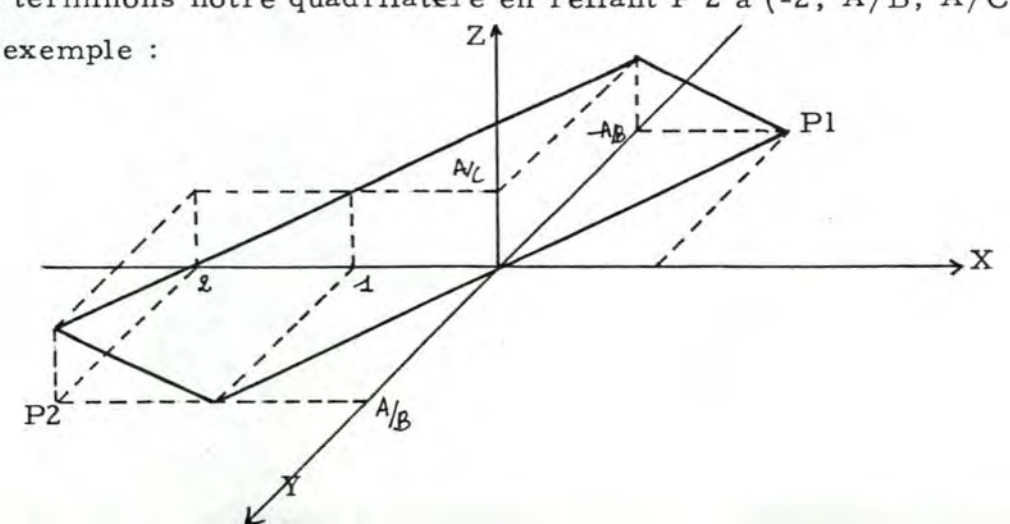
Pour le moment nous n'avons considéré que les équations où tous les coëfficients (A, B, C et D) étaient non-nuls. Voyons maintenant comment nous pouvons représenter une équation du type AX + BY + CZ = 0. Ce plan passe par le point origine (0,0,0) mais n'a aucune autre intersection avec les axes Nous allons nous baser sur la droite d'intersection du plan avec le plan horizontal (Z = 0). Donc l'équation est AX + BY = 0. Prenons deux points (1, -A/B, 0) et (-1, A/B, 0). Par exemple :



Pour donner une idée de l'inclinaison du plan nous avons choisi de relier P 1 au point du plan debout qui a la même coordonnée (-A/B) sur l'axe Y : (O, -A/B, A/C).

Nous terminons notre quadrilatère en reliant P 2 à (-2, A/B, A/C)

Par exemple :



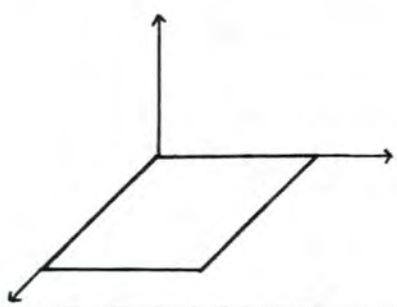
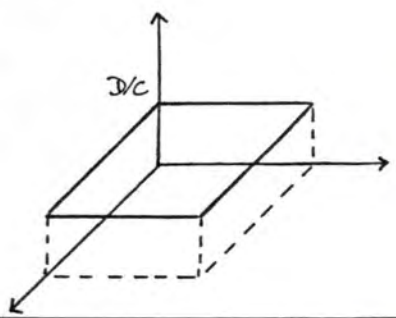
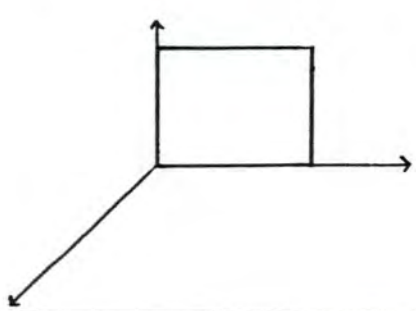
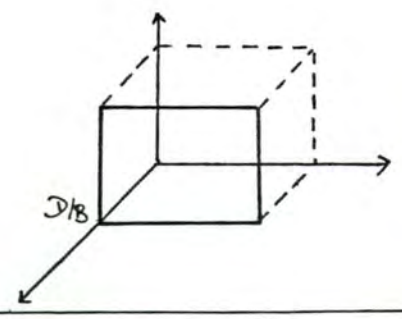
Comme pour les plans AX + BY + CZ = D, nous "doublons" la figure si elle est située sous le niveau horizontal. Nous changeons également d'orientation s'il y a un problème de vision.

Il reste à examiner tous les types de plan où les coëfficients A, B et C ne sont pas tous non nuls.

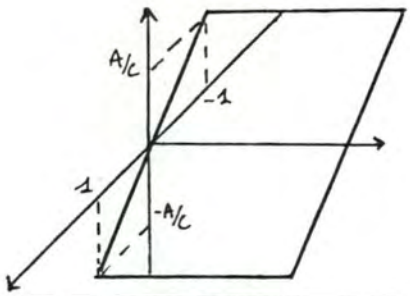
Ces représentations sont assez simples car si un (ou deux) coëfficient(s) est (sont) nul(s), le plan est parallèle à l' (aux) axe(s) correspondant(s) au(x) coëfficient(s) nul(s).

Nous verrons plus tard qu'il peut aussi y avoir certains problèmes de représentation.

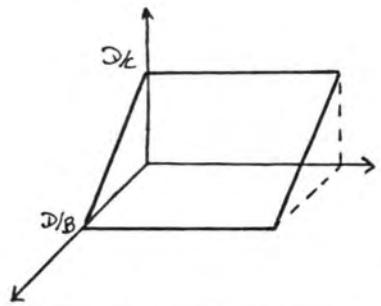
Voyons un exemple de chacun des 16 cas (selon que les coëfficients sont nuls ou non.

0X + 0Y + 0Z = 0 Cette équation est complètement indéterminée. Tous les points (X, Y, Z) en sont solution.	0X + 0Y + 0Z = D Cette équation est impossible, aucun point n'en est solution.
0X + 0Y + CZ = 0 	0X + OY +CZ = D 
0X + BY + 0Z = 0 	OX + BY + 0Z = D 

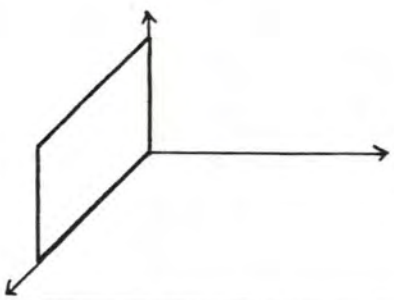
$0X + BY + CZ = 0$



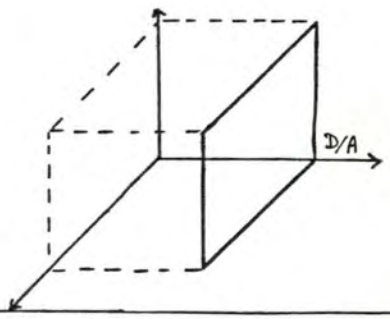
$0X + BY + CZ = D$



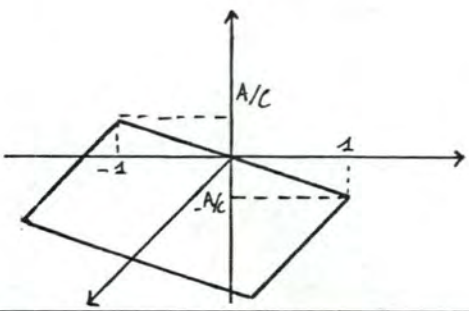
$AX + 0Y + 0Z = 0$



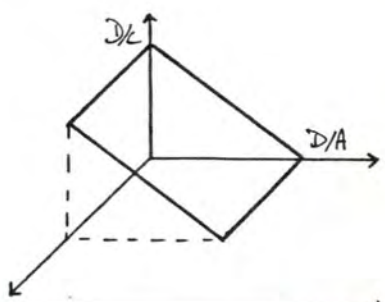
$AX + 0Y + 0Z = D$



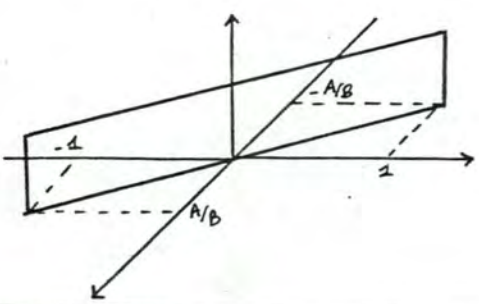
$AX + 0Y + CZ = 0$



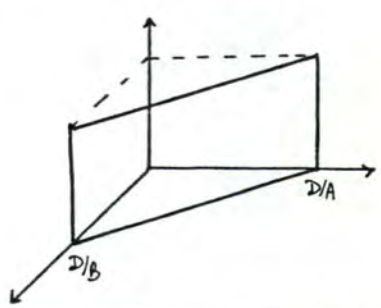
$AX + 0Y + CZ = D$



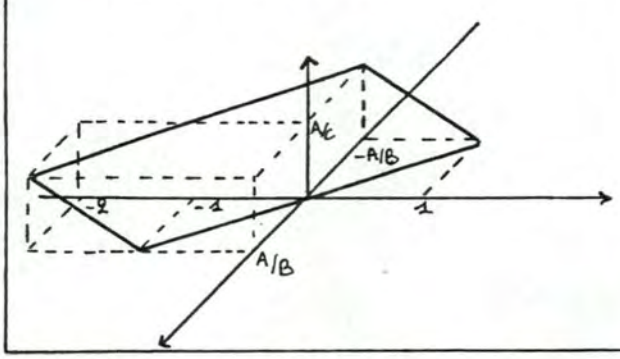
$AX + BY + 0Z = 0$



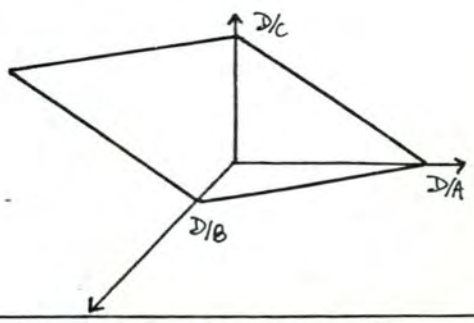
$AX + BY + 0Z = D$



$AX + BY + CZ = 0$

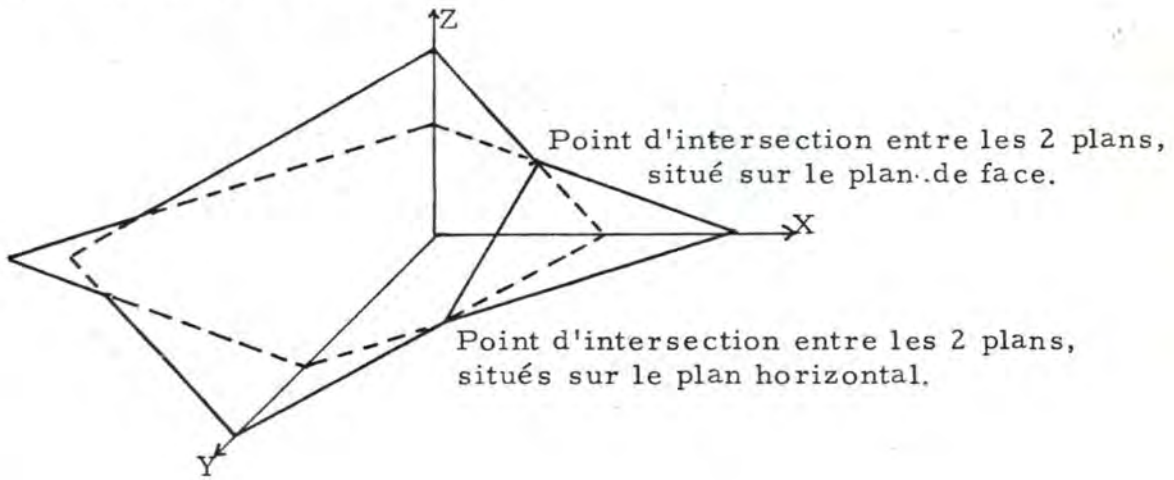


$AX + BY + CZ = D$

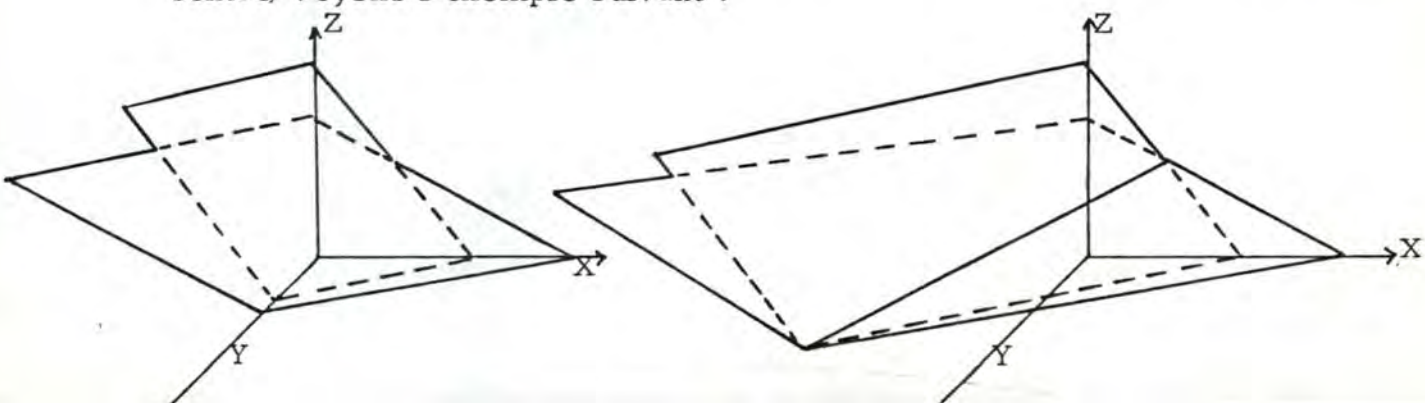


2.2.2. Représentation simultanée de deux de ces équations.

Nous avons vu que, sauf indétermination (0 = 0) ou impossibilité (0 ≠ 0), nous pouvons représenter chaque équation à 3 inconnues par 1 plan. Pour ce faire nous nous basons en général sur les droites d'intersection de ce plan avec les plans horizontal, debout et/ou de face.
Nous devons d'abord déterminer si ces plans sont confondus, parallèles ou sécants. Dans ce dernier cas, nous présenterons la droite d'intersection : notre idée de départ est encore de partir de 2 points faciles à situer. Nous traçons un segment de droite dont les extrémités sont respectivement les intersections entre les 2 droites sur le plan horizontal et et entre les 2 droites sur le plan de face.
Le cas le plus simple est le suivant :



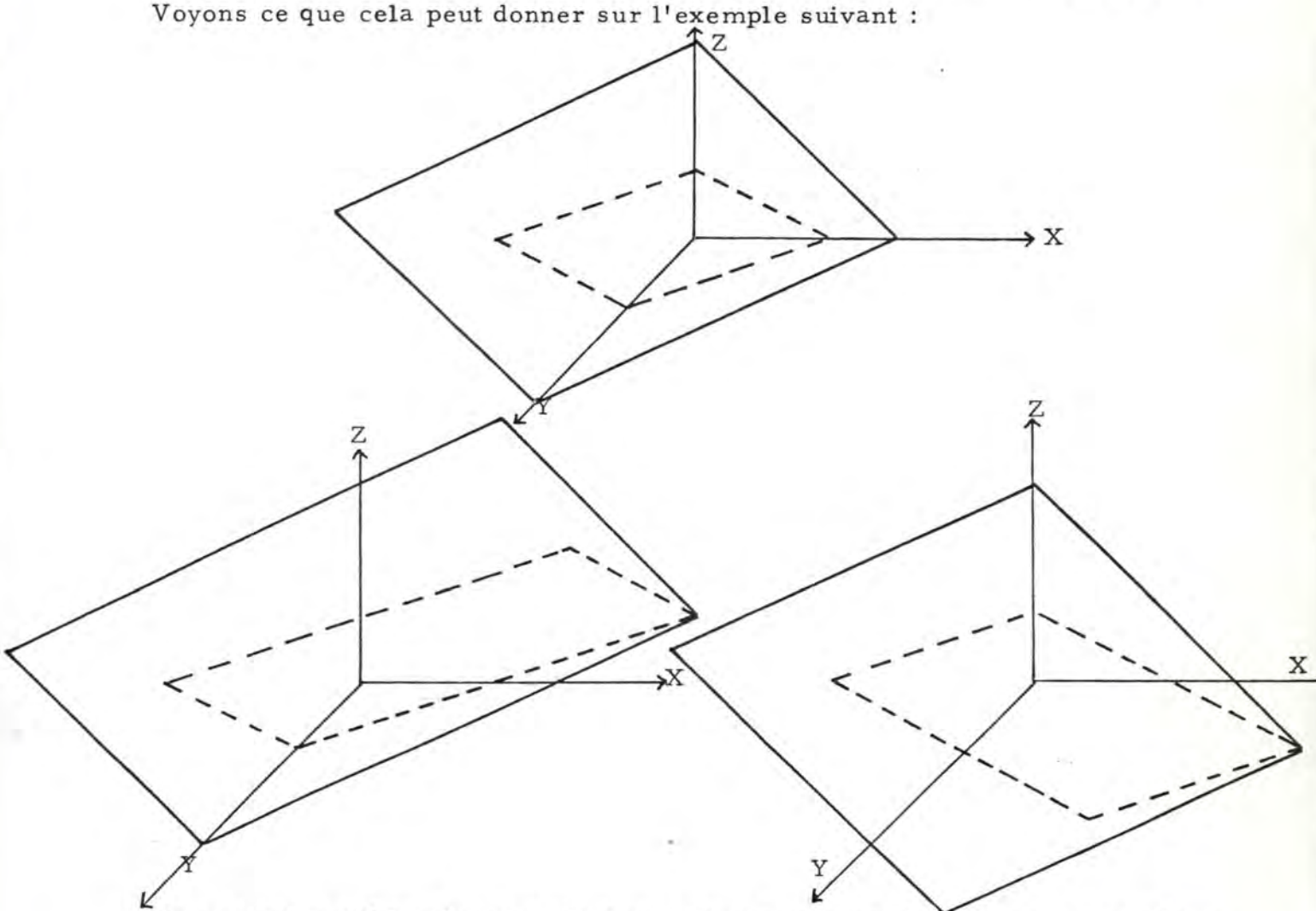
Remarque : Sur ce schéma, nous avons représenté les traits cachés par des pointillés. En pratique le système ne fonctionne pas encore comme cela, même si c'est très utile (voir chapitre des améliorations possibles); chaque plan sera représenté en traits continus partout, mais dans une couleur spécifique.
Toutefois déterminer la droite d'intersection ne sera pas toujours aussi simple car il se peut que les deux points que l'on cherche ne soient pas visibles sur le schéma. Dans ce cas il faut modifier la portion de plan représentée; Voyons l'exemple suivant :



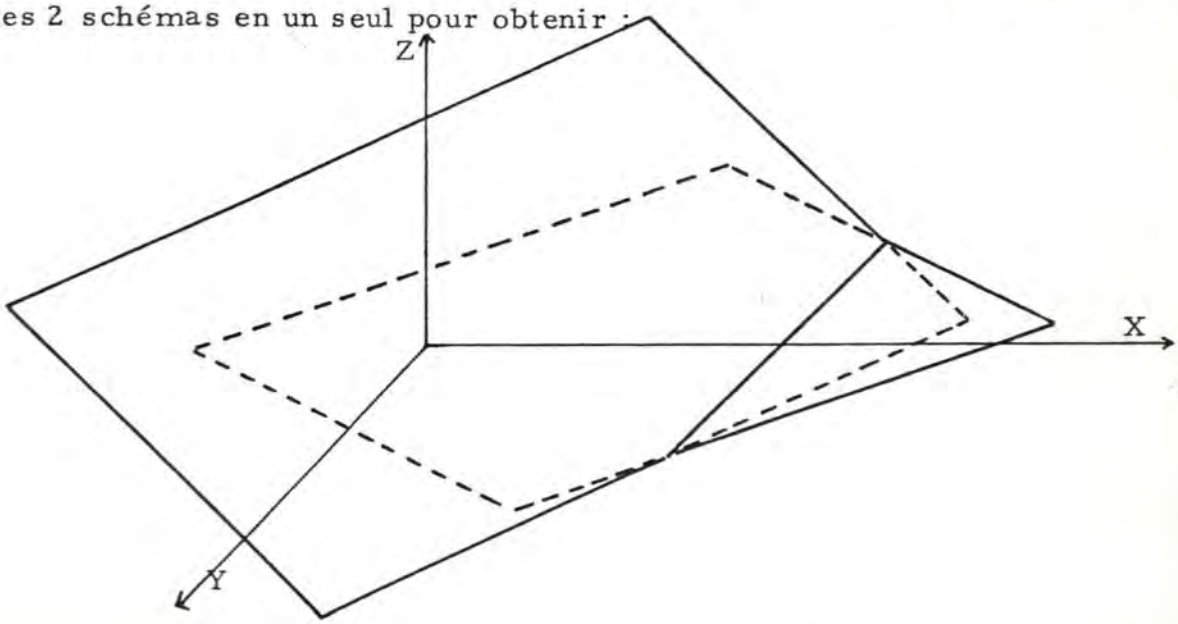
Pour situer le point d'intersection sur le plan horizontal, nous avons prolongé les deux droites d'intersection avec ce plan. (Rem. : Si ces deux droites ont des pentes presqu'égales, le point d'intersection sera situé très loin. Il est alors possible que le dessin ne soit pas explicite. Nous y reviendrons dans le chapitre des améliorations possibles).

Dans d'autres cas nous pourrons prolonger les 2 droites dans le plan vertical, voire même les 4 droites d'intersection avec les plans horizontal et de face.

Voyons ce que cela peut donner sur l'exemple suivant :



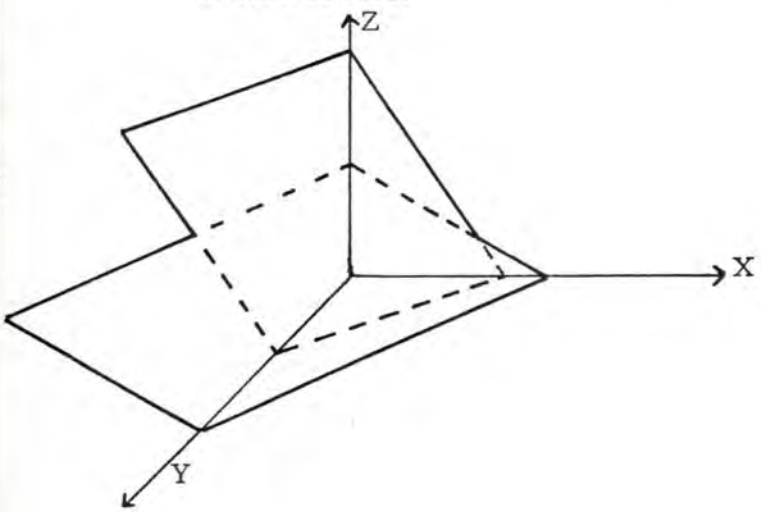
Nous avons situé les 2 points d'intersection que nous cherchions; nous allons intégrer ces 2 schémas en un seul pour obtenir :

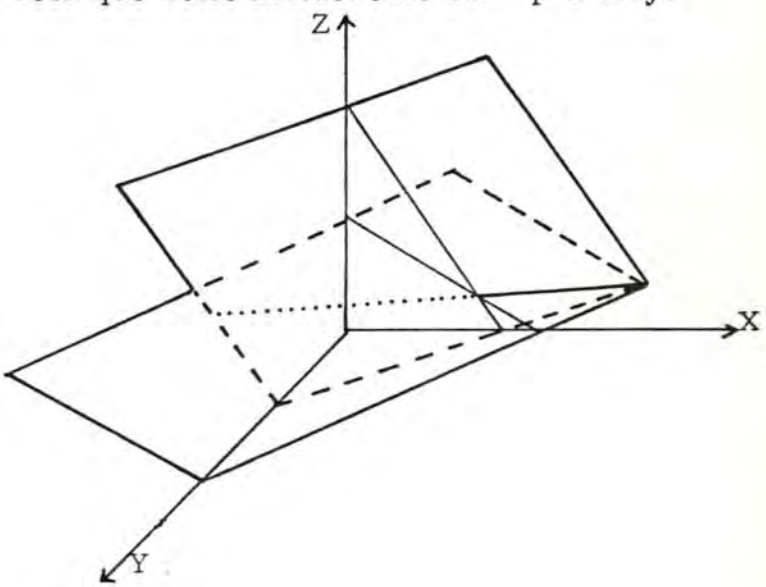


Le principe que nous avons utilisé est le suivant :

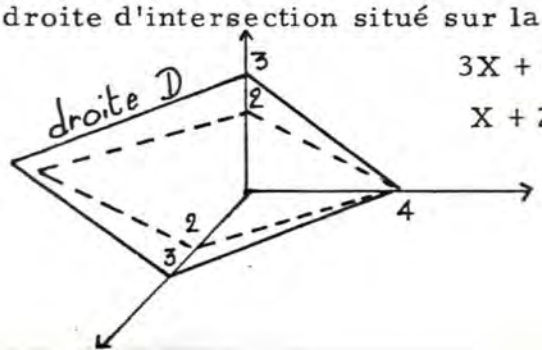
1. Déterminer le point d'intersection dans le plan horizontal.
2. "Prolonger" le plan dans le plan horizontal vers ce point (cela revient à calculer 2 nouveaux sommets si le point n'est pas sur le schéma. Sinon, nous ne modifions pas la portion du plan).
3. Déterminer le point d'intersection dans le plan de face.
4. "Prolonger" le plan dans le plan de face vers ce point. (N. B. Cette prolongation doit évidemment se faire sur base du parallèlogramme éventuellement modifié par la première prolongation).
5. Relier les 2 points d'intersection.

Dans l'exemple suivant nous allons voir que cette méthode ne sera pas toujours satisfaisante.





En reliant les 2 points d'intersection nous ne montrons pas une grande partie de la droite d'intersection. A notre avis il faut prolonger ce segment de telle manière qu'il couvre au moins une des 2 portions de plan représentée. (.....)
Le point 5 du principe est à revoir : il faut d'abord vérifier si les points d'intersection sont sur les droites-limites d'un parallèlogramme ; sinon il faut calculer une nouvelle extrémité du segment, qui remplacera celle qui était "au milieu" du schéma.
De plus, ce principe même corrigé n'est pas valable dans tous les cas :
Il se peut que le point d'intersection dans le plan horizontal et celui dans le plan de face soient confondus.
Dans ce cas nous élèverons le segment de la droite d'intersection, de ce point vers le point de la droite d'intersection situé sur la droite limite "la plus haute".



$3X + 4Y + 4Z = 12$

$X + 2Y + 2Z = 4$

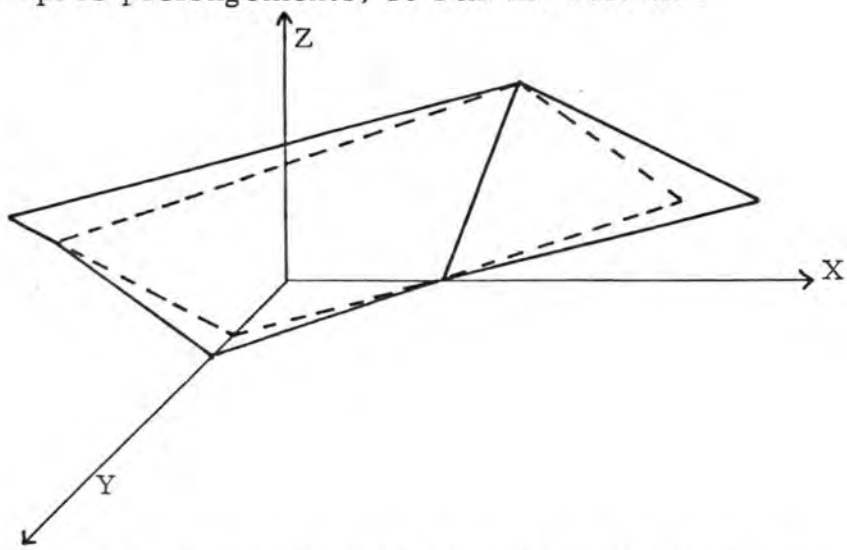
Nous avons un point d'intersection. Nous allons en chercher un second, situé sur la droite D (ou prolongement). Pour ce faire nous déterminons l'équation de la droite d'intersection et comme nous connaissons la "hauteur" de la droite nous pouvons trouver le point que nous cherchons.

Pour notre exemple :

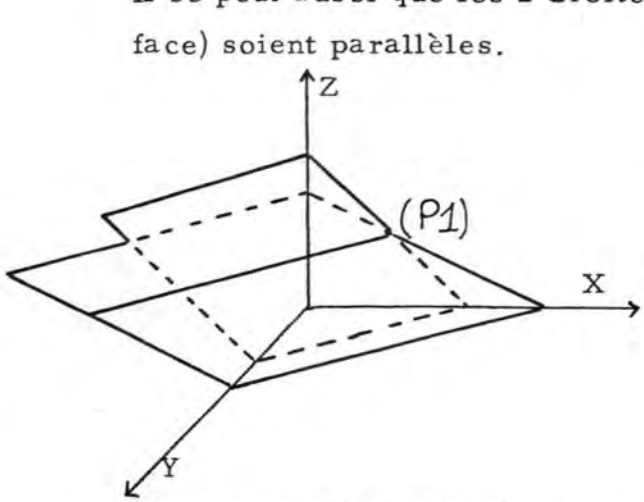
$$3X + 4Y + 4Z = 12 \quad \text{ou} \quad 3X + 4Y + 4Z = 12$$
$$X + 2Y + 2Z = 4 \qquad\qquad 2Y + 2Z = 0$$

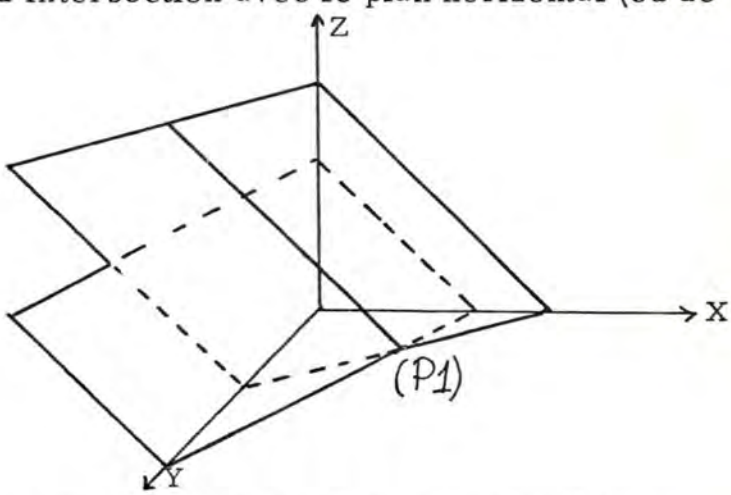
Comme Z = 3 , Y = -3 et X = 4. Le point que nous cherchons est (4, -3, 3).

Nous obtenons , après prolongements, le schéma suivant :



Il se peut aussi que les 2 droites d'intersection avec le plan horizontal (ou de face) soient parallèles.

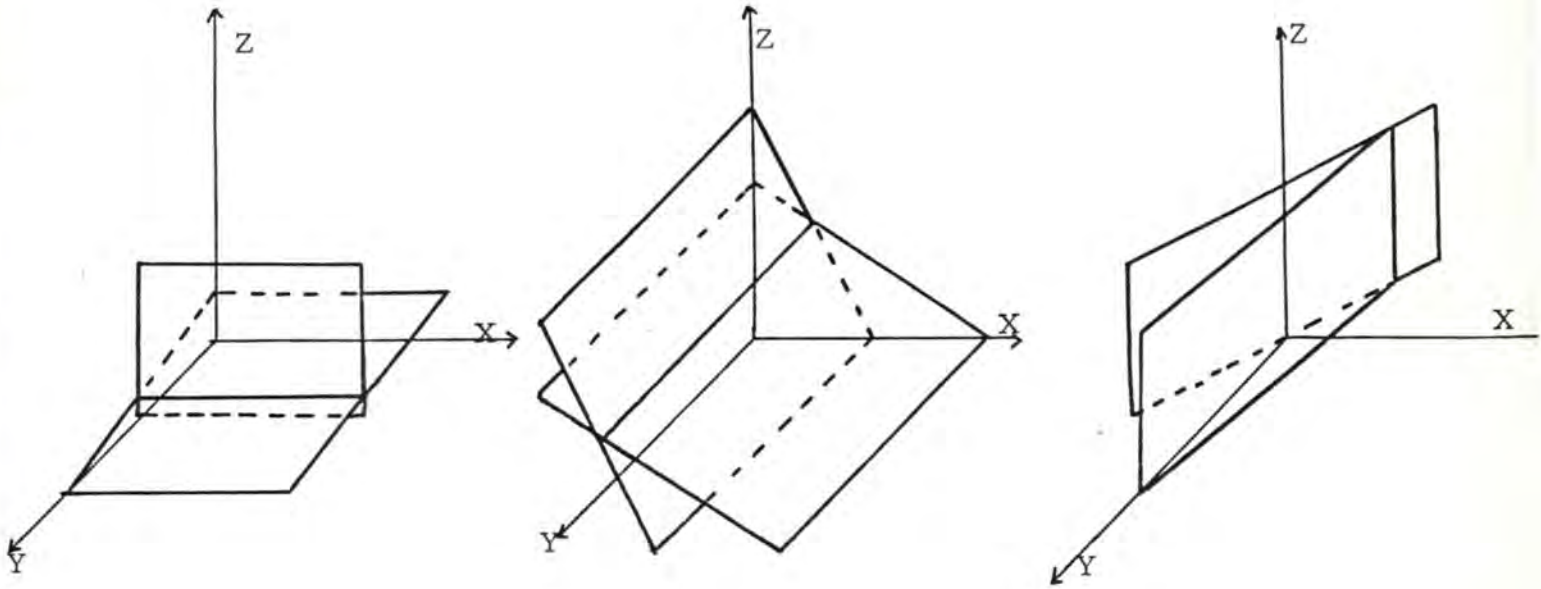




Dans ce cas la droite d'intersection est parallèle aux 2 droites parallèles. Nous traçons donc un segment du point (P 1), parallèlement à ces droites vers la droite limite du plan la plus éloignée.

Il existe aussi d'autres cas particuliers: lorsqu'une même variable n'apparait dans aucune des deux équations, nous savons que ces deux plans sont parallèles à un même axe et que donc, s'il y a une droite d'intersection elle sera parallèle à cet axe.

Voyons un exemple de chaque type :



Pour ces 3 types de système le principe est le suivant :

1. Calculer le premier point d'intersection (situé respectivement dans les plans debout, de face et horizontal).
2. "Prolonger" les 2 plans dans le sens du plan correspondant vers ce point.
3. Calculer le second point.
4. Prolonger les plans vers ce point.
5. Relier les 2 points.

Il existe enfin une dernière classe de système : celle où les 2 équations ne contiennent qu'une seule et même variable. Dans ce cas les deux plans sont parallèles aux 2 mêmes axes et ils sont donc soit confondus, soit parallèles. Il n'y a dons jamais de droite d'intersection dans ces cas.

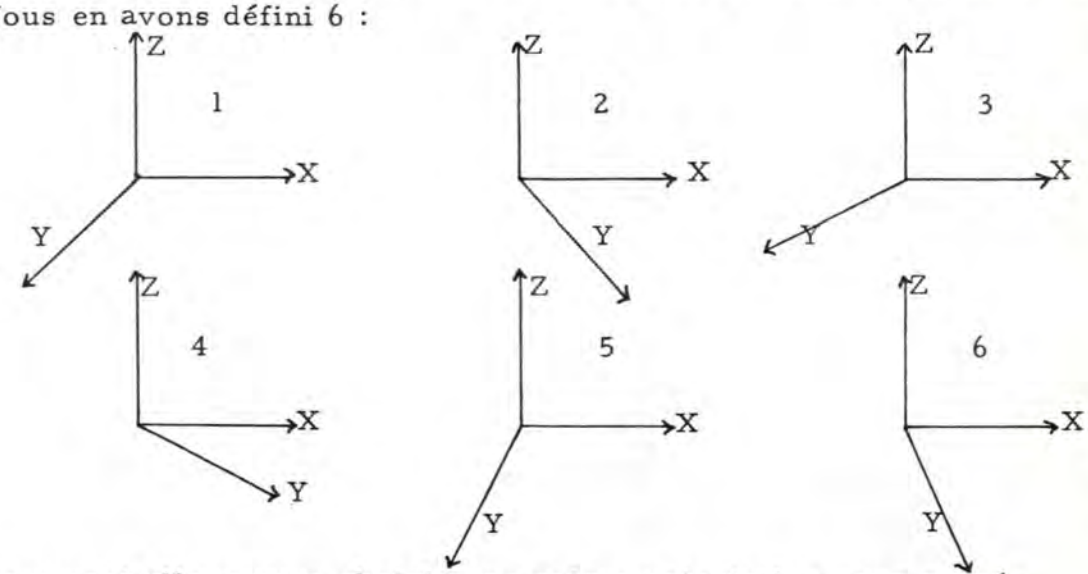
Ce que nous devons faire en général pour représenter simultanément 2 plans peut donc se resumer comme suit :

1. Déterminer le "type" du système de 2 équations, c.a.d. vérifier s'il n'y a pas d'équation impossible ou indéterminée ou encore si une ou deux des variables ne sont pas simultanément absentes des 2 équations.
2. Selon ce type, étudier les coëfficients de manière à déterminer si les plans sont confondus, parallèles ou sécants. (dans ce troisième cas nous appliquerons les méthodes décrites plus tôt pour déterminer les portions de plan à représenter et les extrémités du segment de la droite d'intersection).
3. Il reste alors à représenter ces 2 plans sur un même graphique : Nous devons pour cela nous fixer une unité et un système d'axes de référence communs.

Nous choisirons une unité telle que tous les points que nous voulons représenter (sommets des parallèlogrammes, extrémités du segment d'intersection) soient visibles à l'écran.

Il se peut qu'en procédant ainsi la projection d'un point sur l'un ou l'autre axe ne puisse être représentée sur le dessin. Toutefois nous avons adopté cette méthode afin de représenter les plans par des schémas aussi grands que possibles. Choisir l'unité de telle manière que les 3 projections de chaque point soient visibles imposait une contrainte qui dans pas mal de cas aurait eu comme effet de réduire la taille du schéma.

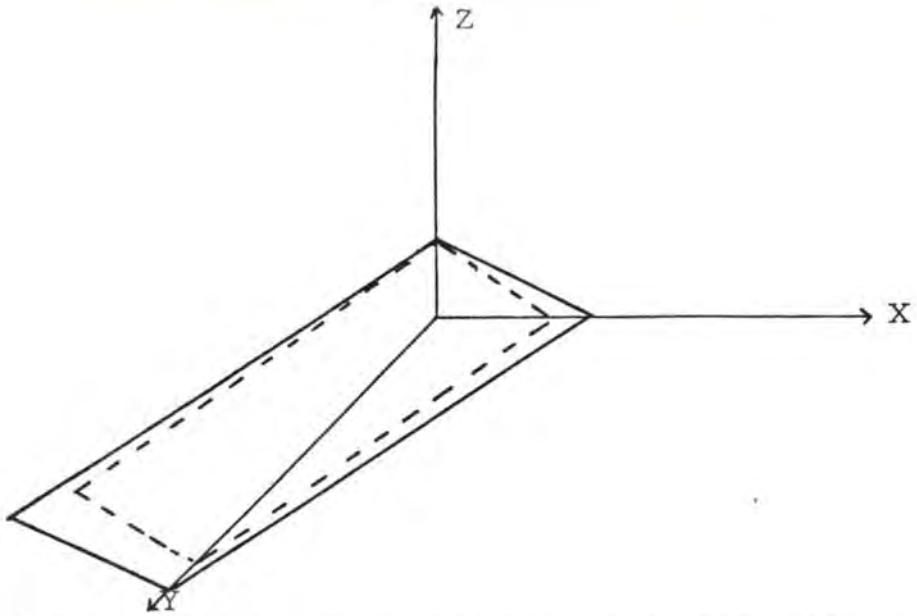
En ce qui concerne le système de référence nous avons vu que nous n'utilisons pas toujours le même (voir point 2.2.1). Si nous avions un problème de représentation dans notre système de départ nous passions à un autre. Cependant il se peut qu'un plan ne puisse être visualisé dans un système de référence et que le second ne puisse l'être dans cet autre. Ainsi 2 systèmes d'axes ne sont pas suffisants. Nous en avons défini 6 :



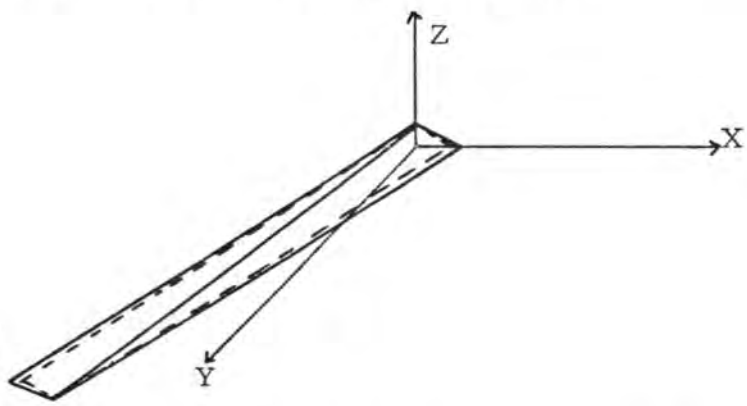
Comment allons nous choisir une même orientation pour représenter 2 plans?

Lorsque nous calculerons les sommets pour représenter un plan, nous vérifions s'il y a des orientations à proscrire et nous les mémorisons. Nous faisons cela pour les 2 plans et nous sélectionnons comme orientation commune la première (dans l'ordre 1 à 6) qui n'est interdite pour aucun des 2 plans.

Comme annoncé au point 1.1.1, nous présentons ici un cas où la superposition de 2 plans donne un schéma assez "brouillon". Cherchons l'intersection entre les 2 plans suivants :



Le point d'intersection dans le plan de face est déja sur le schéma (sur l'axe X 3). Par contre, pour "trouver" celui sur le plan horizontal, nous devons prolonger très loin car les deux droites sont presque parallèles. Ainsi nous choisissons une unité assez petite pour permettre de situer ce point sur le schéma et à cause de cela nous obtenons un schéma du genre :



2.2.3. Représentation de 3 équations à 3 inconnues.

Cette représentation doit permettre de visualiser si le système est impossible, indéterminé (d'ordre 1 ou 2) ou s'il admet une solution unique.
En général nous avons donc à représenter simultanément 3 plans sur un même schéma. Si ces plans sont sécants deux à deux, nous représentons également les 3 intersections. Donc nous aurons pas mal de lignes différentes sur le shéma et il sera pratiquement nécessaire de recourir à l'utilisation des couleurs pour permettre une bonne représentation.
Pour définir les portions de plan à dessiner pour que le schéma soit clair nous aurons recours aux mêmes méthodes de prolongement qu'au point 2.2.2.
Le problème des unités et des orientations est résolu également de la même façon.

Il reste une remarque à faire : si nous devons présenter le "che minement graphique" à l'utilisateur, nous allons lui présenter une vue à l'écran du système modifié à chaque étape. Ce n'est qu'à la fin de ces étapes que nous connaîtrons la solution. Si celle-ci est unique, elle est située à l'intersection des 3 droites d'intersection 2 à 2. Toutefois il n'est pas dit que ce point soit représenté sur chaque schéma. Pour en être sûr,il faudrait toujours prolonger les segments des droites d'intersection vers ce point, s'il est situé "hors du schéma". Il faudrait donc connaître ce point dès le départ et ainsi calculer immédiatement la solution, avant de présenter les différentes étapes.
Nous avons choisi de ne pas le faire (et donc de présenter certains schémas où l'on ne peut que "deviner" la solution). Nous ne trouvons pas cela grave, car l'utilisateur voit évoluer le système et il pourra situer de mieux en mieux la solution.

2.3. AFFICHAGES A L'ECRAN.

Nous avons expliqué (voir point 1.2) les différents écrans proposés à l'utilisateur. Certains donnaient des remarques d'utilisation, des explications ..., d'autres proposaient des choix.
Le but de cette partie est de présenter les informations à l'écran et le cas échéant de saisir une réponse valide (une procédure de validation sera nécessaire).

2.4. INTRODUCTION DES SYSTEMES ET DES REPONSES.

Nous désirons permettre à l'utilisateur d'introduire son système ou sa réponse comme s'il l'écrivait sur une feuille. Voyons les conventions que nous avons fixées d'abord pour l'introduction des systèmes, ensuite celle des réponses.

2.4.1. Introduction des sytèmes.

- Un système d'équations est composé d'un ensemble d'équations. Comme dans la pratique, les exercices demandés sont le plus souvent limités à l'ordre 3x3 (éventuellement de temps à autre un 4x4), nous avons décidé de ne pas dépasser 5 équations à 5 inconnues.
- Chaque équation comprend d'une part une ou plusieurs inconnues (avec éventuellement un coëfficient et un signe) et d'autre part un terme indépendant. Ces deux parties de l'équation se situent respectivement à gauche et à droite d'un signe "="
- Chaque inconnue est représentée par la lettre "X", suivie d'un indice, qui est un nombre entier compris entre 1 et 5 (puisqu'on n'admet que 5 inconnues au

maximum). Les inconnues peuvent être introduites dans un ordre quelconque.

- En principe, à chaque inconnue sont associés un coëfficient et un signe. Signalons que pour simplifier, nous n'admettons pas de signe "*" entre le coëfficient et l'inconnue. De plus ce coëfficient peut être omis : dans ce cas il est assimilé à 1.
  Exemple : " X 1" équivaut à " 1 X1 "
  Le signe peut être "+" ou "-", mais nous n'admettons qu'un seul signe devant un coëfficient.
  Exemple : " + - X 1 " n'est pas admis.
  En début de ligne , le signe peut ne pas être précisé il est alors "+".
  Exemple " 3 X 1 + 2 X 2 ..." correspond à " + 3 X 1 + 2 X 2 ..."
  Cette convention est également appliquée s'il n'y a pas de signe devant le terme indépendant (c'est à dire après le signe "=").

- Les coëfficients de même que le terme indépendant peuvent être de trois types: entiers, fractionnaires et décimaux. Les plus habituels pour les élèves sont bien sûr les entiers, mais comme nous voulions manipuler des fractions afin de présenter le résultat sous cette forme, nous acceptons également ce type de coëfficient. Enfin nous acceptons également des coëfficients décimaux.

  - Chaque fraction se compose d'un numérateur et d'un dénominateur (qui sont tous deux des nombres entiers), séparés par la barre de fraction "/". Son signe doit toujours être placé devant le numérateur(" - 3/4 " et non " 3/-4 "). La fraction doit se trouver devant l'inconnue ("1/2X 1" et non pas " X 1/2 ").

  - Chaque coëfficient décimal se compose d'une partie entière et d'une partie décimale, séparée au choix par un point (".") ou une virgule (",").
    ("1.2 " équivaut à " 1,2 ")
    Si la partie entière est omise, elle sera assimilée à 0. (".2" équivaut à " 0.2").

- A ces conventions de forme nous rajoutons 4 conventions d'écriture
  - L'utilisateur peut insérer des blancs où bon lui semble.
  - La fin de chaque équation est signalée par un "RETURN".
  - La fin du système est signalée
    - Soit par un "RETURN" en début de ligne ("RET").
    - Soit par l'introduction d'une 5e équation (une sixiéme ne sera pas acceptée).
  - Le caractère "←" permet d'effacer la ligne en cours d'introduction. Il faut alors réintroduire complètement la ligne. Cela peut être assez lourd si nous avons déja introduit presque toute la ligne et s'il n'y a

par exemple qu'un seul caractère à modifier. Nous avons pris cette décision afin de ne pas compliquer trop la fonction d'effacement. Pour remédier à cela nous avons recours à une solution qui, si elle n'est pas la plus élégante n'en est pas moins pratique et simple à implémenter: elle consiste à réintroduire la valeur que l'on veut modifier.

Exemple : L'utilisateur a introduit "2/3 X 1 + 2X 2 " au lieu de "2.3 X 1 + 2X 2". Admettons qu'à ce moment il se rende compte de son erreur. Il peut : soit presser la touche "←" : toute la ligne sera effacée et il devra la réintroduire entièrement. (avec nouveau risque d'erreur); soit rajouter à ce qu'il a déja introduit "+ 2.3 X 1". Comme c'est la dernière valeur du coëfficient qui sera enregistrée son erreur est corrigée.

En plus cette fonction ne permet pas d'effacer une ligne autre que la ligne courante, Pour le faire il faut attendre que tout le système soit enregistré: nous demandons alors à l'utilisateur s'il veut modifier l'une ou l'autre ligne. Il était bien sûr possible de prévoir les modifications d'une autre façon : par exemple, un curseur qui se déplace de ligne en ligne et de caractère en caractère, pour permettre d'effacer et/ou de remplacer des caractères n'importe où. Encore une fois, nous n'avons pas voulu compliquer le système, au prix, évidemment d'une utilisation moins souple.

Nous avons vu pourquoi nous voulons toujours éviter un passage inutile vers le mode réel (nous parlons de mode réel ou de mode flottant si nous sommes obligés de travailler avec des réels. Nous parlons de mode fraction si les coëfficients peuvent être représentés par des fractions). Nous devons donc être prudent: lorsque nous effaçons ou corrigeons une ligne, il se peut que celle-ci contienne un coëfficient ou un terme indépendant qui ne permette pas l'enregistrement en mode fraction. Nous avons donc été obligés d'enregistrer au moins ce coëfficient en mode réel. Si maintenant nous le modifions (et qu'il n'y a pas d'autre coëfficient de ce genre), il se peut que le mode réel ne soit plus obligatoire. Cela implique que nous devons pouvoir travailler en mode fraction. Il faudra donc veiller à ce que tout passage en mode réel ne soit pas irréversible, tant que l'utilisateur n'a pas complètement terminé d'introduire son système.

- Il est évident qu'à un moment donné l'utilisateur peut violer une de ces conventions en tapant un caractère qui n'a pas de sens dans le contexte.
Dans ce cas nous n'imprimons pas ce caractère à l'écran et signalons à l'utilisateur qu'il vient de commettre une erreur: nous l'invitons à réintroduire un caractère. A cet effet nous lui donnons la liste des types de caractères qui sont admissibles dans la situation où nous sommes.

Exemple : L'utilisateur a déjà introduit " 2 X 1 " . Si à ce moment il tape "/", il y a une erreur car cela n'a aucun sens d'après nos conventions. Nous n'imprimons donc pas ce caractère et nous envoyons un message pour signaler que les seuls caractères qui peuvent suivre sont "+", "-", ou "=".
Remarque: Comme il peut également introduire un espace ou la flèche d'effacement à tout endroit, nous ne le rapellons pas dans le message, puisque ces deux caractères sont toujours valides.

2.4.2. Introduction des réponses.

Comme nous l'avons signalé dans le scénario, nous nous limitons dans le cas de systèmes indéterminés à vérifier l'ordre d'indétermination. Nous n'avons donc qu'à saisir les solutions uniques.
Les conventions sont les suivantes:

- Chaque ligne a la forme X indice = valeur.
  L'indice doit être compris entre 1 et le nombre d'inconnues dans l'exercice. Nous afficherons "X 1 = " sur la première ligne et l'utilisateur devra complèter par la valeur qu'il a trouvée. Cette valeur peut avoir la même forme qu'un coëfficient (voir description au point 2.4.1 : Introduction des systèmes).
- La fin de chaque ligne est exprimée par "RETURN" (ainsi la fin de l'introduction est marquée par le "RETURN" terminant la dernière ligne).
- Les explications concernant les blancs, la touche d'effacement et les messages d'erreur sont les mêmes que pour l'introduction des systèmes.

2.5. LA RESOLUTION DES SYSTEMES.

Comme nous l'avons signalé les deux contraintes suivantes doivent être respectées :

- Manipuler des fractions.
- Résoudre les systèmes par élimination.

Nous traitons donc les coëfficients comme des fractions. Pour cela nous allons représenter chacun d'eux par deux nombres entiers , un numérateur et un dénominateur. Il y a malheureusement une limite : la capacité maximale des nombres entiers n'est pas très élevée (aux environs de + ou - 32000). Cela implique

que si, à un moment donné, un des deux nombres entiers dépasse cette capacité, les coëfficients ne sauront plus être exprimés ainsi. Nous utiliserons alors des nombres réels(au sens Pascal du terme). Notons que dans ce cas la limite est beaucoup plus élevée mais qu'il n'est rien prévu si la capacité est dépassée.

Dans la mesure du possible nous essaierons toujours d'éviter l'uitilisation des réels, et ce pour deux raisons :

- Nous ne pouvons alors plus présenter les systèmes à l'élève sous la forme qu'il utilise.
- Nous perdons de la précision du fait que le nombre de décimales est bien évidemment limité (ce qui n'arrive pas avec les fractions).

Nous retrouverons ce souci d'éviter le passage inutile en réel dans tout ce qui va suivre.

La première chose que nous ferons avant de résoudre le système sera de le simplifier au maximum.

- Une équation 2/4 X 1 - 8/10 X 2 = 3/9.
  devient 1/2 X 1 - 4/5 X 2 = 1/3.
  Nous simplifions toutes les fractions.
- Une équation 2 X 1 - 4 X 2 = 6
  devient X 1 - 2 X 2 = 3
  Si une équation ne contient que des coëfficients entiers nous simplifions chaque coëfficient par leur P G C D.

Notons qu'il est important de d'abord essayer de simplifier les fractions et ensuite de vérifier s'il est possible de simplifier toute l'équation.

- Par exemple : 4/2 X 1 + 16/4 X 2 = 2
  simplifions les fractions : 2 X 1 + 4 X 2 = 2
  simplifions l'équation : X 1 + 2 X 2 = 1

Il y a deux raisons à cette simplification de départ:

- Dans la majorité des cas les élèves essaient de simplifier les équations et/ ou les fractions afin de ne pas avoir à effectuer des calculs sur des nombres trop élevés.
- Mieux nous parvenons à simplifier au départ, plus nos nombres sont petits et moins nous risquons de devoir traiter des nombres réels.

Pour réaliser ces simplifications nous devrons chercher le P G C D soit de tous les coëfficients de la ligne, soit du numérateur et du dénominateur.

Comment procède l'élève quand il cherche un P G C D ? En général, il essaie

de diviser chacun des nombres par 2, 3 et 5. Lorsqu'il a fait cela il compare les nombres qui lui restent et si une simplification semble évidente, il l'effectue. Sinon il arrête là et il n'a pas nécessairement simplifié par le P G C D.

- Voyons un exemple : soit l'équation 442 X 1 + 238 X 2 = 0. L'élève divise par deux : 221 X 1 + 119 X 2 = 0. Il essaie 3 et 5. Cela ne marche pas. En fait il est encore possible de diviser par 17 mais ce facteur est très difficile à trouver. L'équation entièrement simplifiée devient 13 X 1 + 7 X 2 = 0. Comme cela nous évitera peut-être de passer en réel, nous effectuons cette simplification, même s'il y a beaucoup de chances que l'élève ne le fasse pas. Cependant nous enverrons un message à l'élève pour lui signaler que nous avons divisé par un P G C D difficile à trouver. Et ce pour 2 raisons :
  1. Si l'élève a résolu l'exercice sans simplifier, il risque de trainer le facteur "17" dans tous les calculs, alors qu'il aurait pu se simplifier très fort la tâche. Donc, pour ne pas trop le frustrer malgré tout, nous lui envoyons ce message pour signaler que la simplification n'était pas évidente.
  2. Il se peut aussi que si nous ne lui disons pas quel était le P G C D, l'élève ne se rende pas compte de la manière dont nous sommes passés d'une équation à l'autre.

Dans quels cas envoyons nous ce message? Après avoir cherché le P G C D, nous essayons de le diviser succesivement par 2, 3 et 5. Lorsqu'il n'est plus possible de le faire, nous avons comme résultat le facteur du P G C D qui ne contient pas de puissances de 2, 3 ou 5. C'est précisément ce facteur qui est difficile à trouver par l'élève. Le professeur estimera, selon ses élèves, quel est le plus petit facteur que "normalement" ses élèves ne trouveront pas. Il trouvera peut-être que dès qu'un facteur autre que 2, 3 ou 5 intervient dans le P G C D il faut envoyer le message. Au contraire il pourra penser que les facteurs 7 et 11 (par exemple) sont encore faciles à trouver. Ainsi, selon son idée il fixera un maximum admis pour ce facteur restant. Selon que le facteur restant est supérieur ou non au maximum défini par le professeur, nous envoyons ou non le message.

Après cela nous devons effectuer l'élimination, c. à. d. que nous allons éliminer les variables dans certaines équations : en général l'élève suit la méthode suivante : à chaque étape

1. Il regarde s'il n'y a pas une équation du type $AX_i = B$, ce qui permet de remplacer X_i par sa valeur B/A dans toutes les autres équations.

2. Si non, il va éliminer une variable, c'est à dire que par des multiplications et des soustractions d'équations il va faire disparaître une des inconnues de certaines équations.

Pour la sélection de la variable à éliminer nous lui proposons la méthode suivante: nous éliminerons toujours l'inconnue qui apparaît dans le moins d'équations. (Si plusieurs inconnues apparaissent un même nombre de fois, nous prenons parmi celles-ci, celle dont l'indice est le plus petit).

Nous avons retenu cette méthode parce qu'elle présente un critére simple et objectif et qu'elle minimise le nombre d'éliminations à opérer (sans tenir compte de la complexité des calculs).

Voyons un exemple :

$$\begin{cases} 2X1 + X2 - 3X3 = 2 \\ 4X1 - X2 = 5 \\ X1 + 4X2 + X3 = 2 \end{cases}$$

Nous allons sélectionner X3 puisqu'il est absent de la seconde équation. Il n'y aura ainsi qu'une seule élimination à exécuter.

Il est intéressant de faire comprendre cela à l'élève car cela lui évitera d'effectuer trop de calculs(qui peuvent être source d'erreurs).

Si l'élève n'utilise pas ce raisonnement, il choisira peut être d'éliminer X 1 parcequ'il suit toujours l'ordre croissant ou X2 parce qu'il voit que cette inconnue est facile à éliminer de la deuxième équation.

Ce processus va se poursuivre jusqu'à ce que

- soit l équation a la forme 0 = A. Dans ce cas le système est impossible.
- soit toutes les inconnues ont une valeur déterminée. Le système admet une solution unique.
- soit il n'est plus possible d'éliminer ni de substituer alors que toutes les variables n'ont pas encore une valeur. Ce système est indéterminé. L'ordre d'indétermination correspond au nombre d'inconnues auxquelles on peut donner une valeur arbitraire.

2. 6. VERIFICATION DES REPONSES.

Comme signalé plus haut, cette vérification sera assez sommaire. Elle ne portera en aucun cas sur la démarche, mais uniquement sur la réponse obtenue. Nous demandons donc pour commencer si la solution qu'il a trouvée est impossible, indéterminée ou unique. S'il a déja commis une erreur à ce niveau-là, nous ne lui demandons rien de plus. Nous lui signalons simplement son erreur. (voir Inier 1, Inier 3, Inier 5).

S'il a trouvé le bon "type" de solution, selon les cas :

- si le système est impossible, nous lui confirmons que c'est correct.
- si le système est indéterminé, nous lui demandons l'ordre d'indétermination et nous le comparons avec celui que nous avons trouvé. Selon le résultat nous lui envoyons l'un ou l'autre message (voir Inier4).
- si le système est à solution unique; nous lui demandons d'introduire celle-ci. Nous comparons alors les 2 réponses et selon le cas, nous lui communiquons son résultat (voir Inier 2).

2.7. CREATION D'EXERCICES.

Comme nous n'avons pas eu le temps de discuter avec les professeurs concernés, pour savoir exactement ce qu'ils attendaient, nous avons décidé de développer un petit système très simple.
Nous demandons à l'utilisateur de choisir le nombre d'équations et d'inconnues qu'il souhaite et nous générons les coëfficients correspondants comme des entiers aléatoires compris entre O et 5.
Il est évident que cela ne satisfera pas nécessairement tous les utilisateurs.
Il leur sera toujours loisible de modifier cette routine selon leurs désirs.

2.8. DEMONSTRATION 2 X 2.

1. Présenter la manière dont une équation à 2 inconnues peut être représentée comme une droite :
   - si les 3 coëfficients sont non-nuls, expliquer que deux points situés sur les axes sont solutions de l'équation et montrer que quelques points situés sur la droite qui les joint sont également solutions.
   - montrer que si un des coëfficients est nul, cela constitue un cas particulier: droite parallèle (ou confondue) avec un axe, droite passant par l'origine.
2. Présenter les 3 types de systèmes 2 x 2;
   - système impossible : les deux droites sont parallèles.
   - système indéterminé (ce sera toujours d'ordre 1): les 2 droites sont confondues.
   - système à une solution: X 1 = A, X 2 = B. Montrer que le point d'intersection entre les 2 droites est justement le point (A, B).

2.9. DEMONSTRATION 3 x 3.

Le principe est le même que pour les systèmes 2 x 2. Mais vu le nombre de cas différents à envisager, il y aura une séléction à opérer dans tous les choix possibles pour que la démonstration ne soit pas trop longue.

Ici trois points nous ont semblé intéressants:

1. Familiariser l'élève avec la représentation d'une équation à 3 inconnues.
2. Montrer comment 2 plans peuvent se situer l'un par rapport à l'autre.
3. Distinguer les différents cas correspondant aux 3 types de solution.

1. Nous avons vu qu'il n'y avait pas moins de 16 types de plans différents, selon que les paramètres sont nuls ou non. De plus les plans d'un même type peuvent avoir une représentation légèrement différente, car la position du système d'axes de référence peut changer. Il importe de choisir un nombre d'exemples suffisant pour que l'élève puisse se rendre compte de toutes les conventions de représentation adoptées, mais pas trop élevé pour ne pas le lasser.

   A notre avis il faut insister avant tout sur deux choses :

   - La représentation d'un plan est basé sur la trace de ce plan sur une partie des plans de référence.
   - Lorsque le coëfficient d'une inconnue est nul, le plan est parallèle à l'axe correspondant.

2. Il convient de montrer que 2 plans peuvent être parallèles, confondus ou sécants. S'ils sont sécants, il faut bien indiquer la position de la droite d'intersection.

   Il sera également intéressant de montrer quelques exemples où la droite d'intersection n'est pas évidente à visualiser.

   Par exemple :2 plans situés dans des parties différentes du système de référence; 2 plans passant par l'origine; ...

3. Nous présentons les 3 cas possibles de système et dans chaque cas nous détaillons les différentes possibilités :

   - Le système peut être impossible parce que :
     - soit les 3 plans sont parallèles.
     - soit 2 plans sont confondus et le 3e leur est parallèle.
     - soit 2 plans sont parallèles et le 3e leur est sécant.
     - soit les 3 plans sont tels que leurs intersections deux à deux sont des droites parallèles.

- Le système peut être indéterminé :
  - soit d'ordre 1 (dans ce cas tous les points solution du système sont situés sur une même droite) parce que soit 2 plans sont confondus et le troisième leur est sécant; soit leurs intersections 2 à 2 constituent la même droite.
  - soit d'ordre 2 (tous les points solution du système sont situés sur un même plan) parce que les 3 plans sont confondus.

2.10 COORDINATEUR

Ce rôle est assez classique : en fonction des choix effectués par l'utilisateur en réponse aux menus affichés, il guide l'utilisateur et appelle les différentes parties du programme.

3. SUPPORT GRAPHIQUE 2 X 2 .

Ce module a un double but :

- Représenter les équations du type AX + BY = C.
- Représenter graphiquement les systèmes de 2 équations à 2 inconnues.

3.1 REPRESENTATION GRAPHIQUE D'UNE EQUATION AX + BY = C.

Nous réalisons 3 choses principales :

1. Nous déterminons une unité convenable qui permette de bien "visualiser" la droite à l'écran (c'est à dire que nous voulons que toute intersection de la droite avec l'un ou l'autre axe soit visible).
2. Nous calculons la position des points qui seront les extrémités du segment de droite que nous présentons à l'écran.
3. Nous traçons les axes de référence et le segment de droite défini.

3.1.1. Déterminer une "unité convenable" en fonction des coëfficients A, B et C.

Nous avons pris comme convention de départ d'utiliser la même unité pour les deux axes, afin de donner une idée exacte de la pente.
Nous avons défini la procédure DETERUNITE, dont le but est de déterminer une unité qui assure que les points d'intersection avec les axes (s'il y en a) sont visibles à l'écran.
Nous lui passons 2 paramètres (point d'intersection avec l'axe X, point d'intersection avec l'axe Y) et elle renvoie l'unité cherchée. Par convention si un des paramètres est nul, cela signifie soit que la droite n'a pas de point d'intersection avec l'axe correspondant, soit que l'intersection se situe au point origine. Dans les deux cas, il n'y a pas de contraintes sur l'unité du point de vue de cet axe, puisque soit il n' y a pas de point à y représenter, soit il s'agit du point origine, qui est bien sûr toujours visible.
Nous appelons cette procédure avec des paramètres choisis par la procédure CHOIXUNITE, qui étudie les coëfficients A, B et C pour déterminer le "type" de la droite (sécante aux 2 axes, parallèle à un des axes ...).
Nous distinguons 4 cas selon que A et/ou B sont nuls ou non :

- Si A = B = 0 , il s'agit d'une équation 0 = 0 ou 0 = C, que nous ne pouvons représenter : DETUNITE (0,0) renvoie une unité arbitraire.

- Si A = 0 et B ≠ 0, l'équation est BY = C ou BY = 0. Nous appelons DETUNITE (0, C/B). Si C ≠ 0, l'unité sera telle que le point C/B se situe juste au milieu du demi-axe Y sur lequel ce point est situé. Si C= 0 , l'unité est bien sûr arbitraire (la droite est confondue avec l'axe, toujours vu).

- Si A ≠ 0 et B = 0 le raisonnement est le même (par rapport à l'axe X).

- Si A ≠ 0 et B ≠ 0 l'équation AX + BY = C ou AX + BY = 0.
  Si C ≠ 0 DETUNITE (C/A, C/B) choisit d'abord une unité qui situe le point C/A juste au milieu du demi-axe X. Elle vérifie si cette unité permet de voir C/B sur l'axe Y. Si oui,cette unité est conservée. Si non, elle est recalculée pour que le point C/B soit juste au milieu du demi-axe Y.
  Si C = 0 la droite passe par l'origine et l'unité est arbitraire.

3.1.2. Calcul des extrémités du segment de droite.

(Voir CALCULPTS)

Afin de pouvoir les calculer, nous cherchons d'abord le genre de la droite à laquelle nous avons affaire, si toutefois il s'agit d'une droite. Nous obtenons la structure suivante :

```
Si A = 0      Alors Si B = 0     Alors Si C = 0.
                                        Alors équation indéterminée  (1)
                                        Sinon équation impossible    (2)
                                 Sinon équation (BY = C)             (3)
              Sinon Si B = 0     Alors équation (AX = C)             (4)
                                 Sinon Si C = 0.
                                        Alors équation (AX + BY = 0) (5)
                                        Sinon équation (AX + BY = C) (6)
```

Dans les cas (1) et (2) il n'y a pas de droite à représenter: nous positionnons un indicateur (PASDROITE) à vrai et nous envoyons un message qui explique que l'équation est impossible (voir DRIMPOS) ou indéterminée (voir DRINDET).

Dans tous les autres cas nous allons représenter une droite; pour cela nous allons: (voir GR 22)

- calculer les points (X 1, Y 1) et (X 2, Y 2) qui sont les extrémités de la portion de droite qui représentera l'équation sur l'écran.

- envoyer un message à l'utilisateur pour lui donner des renseignements sur la droite que nous allons lui présenter. Par exemple:

  - "Cette droite est parallèle à l'axe X. Elle coupe l'axe

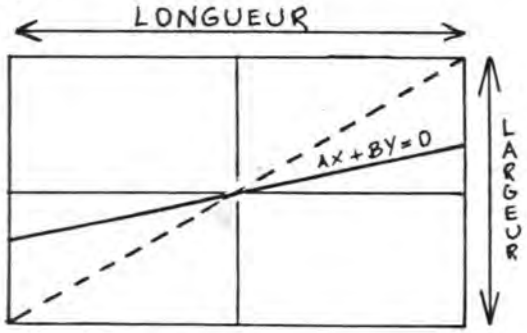
Y au point ..."

- "Cette droite passe par le point origine. La pente vaut ..." etc... etc... (voir DRPARX, DRPARY, DRPP00, DRPP2P).

- positionner les variables E 1 et E 2. Celles-ci nous servent à placer des répères sur les axes.
  Par exemple pour une droite passant par (0, 0) nous positionnons E 1 et E 2 à 1 pour que l'élève voie la distance représentée par une unité.
  Dans ce même esprit nous indiquons également l'échelle (voir procédure ECHELLE).

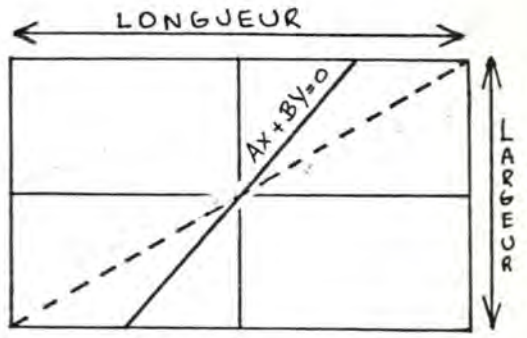
La manière de calculer les extrémités du segment de droite dépend bien sûr du genre de droite.

- Pour une droite parallèle (ou confondue) à l'axe X c'est très simple. Nous allons représenter un segment de droite, partant du bord gauche de l'écran vers le bord droit de l'écran, à hauteur constante C/B. Nous fixons donc Y1 et Y2 à C/B. X1 et X2 sont fixés à -longueur/2 x UN et à longueur/ 2 x UN si longueur/2 représente la longueur du demi-axe X, exprimée en nombre de points lumineux, et UN l'unité.
- Pour une droite parallèle (ou confondue) à l'axe Y, le principe est le même, mais le segment va de bas en haut: X1 = X2 = C/A,
  Y1 = - largeur/2 x UN et Y2 = largeur/2 x UN.
- Pour une droite passant par l'origine, nous déterminons d'abord si la pente de la droite est inférieure ou supérieure à celle de la droite diagonale de l'écran. Ainsi nous savons si les "points limites" de la droite se situent sur les bords inférieur ou supérieur ou gauche ou droit.



$\frac{|A|}{|B|} < \frac{\text{LARGEUR}}{\text{LONGUEUR}}$

Nous donnons à X1 et X2 les valeurs + ou - longueur/ 2 x unité(UN) et nous calculons Y1 et Y2.



$\frac{|A|}{|B|} > \frac{\text{LARGEUR}}{\text{LONGUEUR}}$

Nous donnons à Y1 et Y2 les valeurs + ou - largeur/ 2 x unité(UN) et nous calculons X1 et X2.

- Pour une droite sécante aux deux axes, il y a 6 possibilités de situation pour les extrémités, selon que la droite va du bord gauche aux bords supérieur,droit ou inférieur; du bord droit aux bords supérieur ou inférieur; du bord inférieur au bord supérieur. Nous essayons les différentes possibilités jusqu'à trouver la bonne.

3.1.3. Représentation physique de la droite.

Nous avons vu que dans les 2 cas où nous ne pouvons pas représenter de droite nous positionnons PASDROITE à vrai. Il n'y aura donc de représentation. de droite que si PASDROITE est faux.
Dans ce cas, nous traçons d'abord les axes (voir AXES). Pour ce faire nous tirons profit des outils mis à notre disposition dans "TURTLEGRAPHICS" et qui nous permettent de tracer des droites, d'inscrire du texte dans le graphique ....
Nous faisons ensuite appel à la procédure DROITE qui en fonction des 2 points (X1, Y1) et (X2, Y2) et de l'unité dessine une droite à l'écran. Pour ce faire nous calculons la position physique des points (X1, Y1) et (X2, Y2) sur l'écran par simple déplacement par rapport au centre et nous traçons la droite entre les deux. Nous mémorisons l'unité et les 2 points. Ensuite nous appelons une procédure (voir INTERSECTION) qui place nos repères (E1 et E2) sur le schéma. Nous donnons aussi l'unité (voir procedure ECHELLE).

3.2. REPRESENTATION GRAPHIQUE D'UN SYSTEME 2 X 2.

(Voir procédure SYST22).
Nous commençons par déterminer si les deux droites sont parallèles, confondues ou sécantes. C'est le rôle de CALCULINTER : elle positionne TYPEINTER et renvoie le point d'intersection si les 2 droites sont sécantes. Dans ce cas nous calculons UNITEINTER qui est une unité qui permet de visualiser le point d'intersection.
Après avoir expliqué à l'utilisateur de quel genre de système il s'agissait, nous sélectionnons une unité qui permet de visualiser les 2 droites et leur éventuel point d'intersection.
Nous recalculons ensuite les extrémités pour la ou les droites dont l'unité a été modifiée (procedure RECALCUL). Il s'agit simplement de réexécuter la procédure CALCULPTS sans envoyer les messages et en mémorisant les points trouvés. Il reste alors à tracer les axes et les 2 droites.

4. SUPPORT GRAPHIQUE 3 X 3 .

Nous allons traiter ce problème en 5 temps :

- Nous avons vu que les "points logiques" dans les exemples étaient représentés en 3 dimensions. Nous devrons donc définir la manière de les transformer en "points physiques" en 2 dimensions. Nous parlerons aussi du problème des unités et de la représentation des axes.

-Nous savons que nous ne pouvons pas représenter tous les plans dans n'importe quel système d'axes de référence. Nous modifions parfois la position de l'axe X2 pour permettre une meilleure visualisation. C'est de l'orientation de cet axe que nous allons parler.

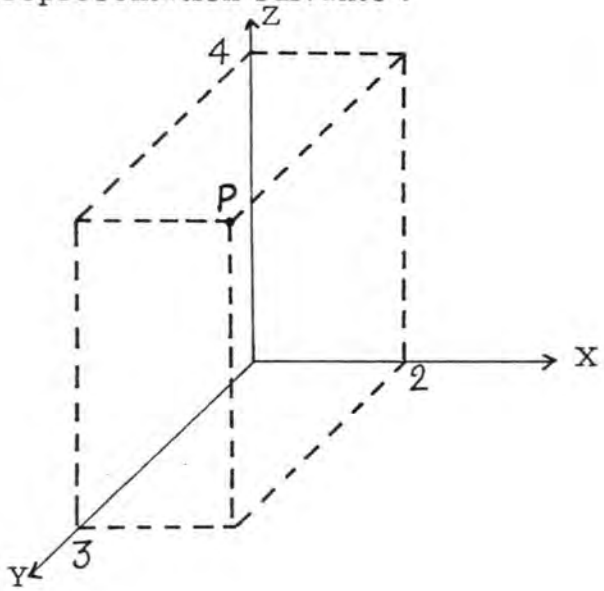
- Nous allons préciser les points que nous choisirons pour représenter les sommets de la portion de plan choisie d'après nos conventions, ainsi que la manière de sélectionner l'orientation du système d'axes.

- Nous verrons ensuite comment superposer 2 plans.

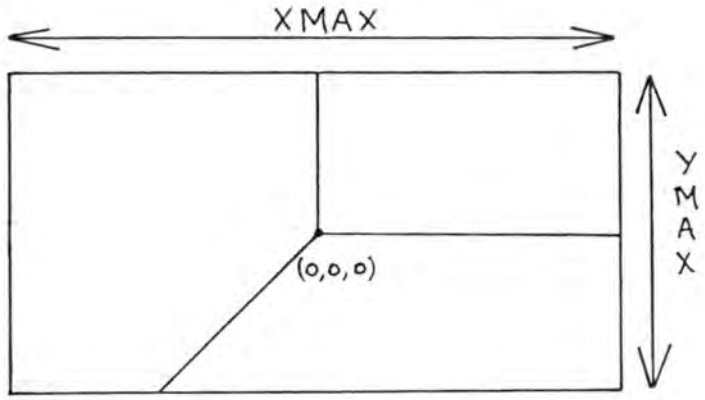
- Nous présenterons alors tout le système 3 x 3.

4.1. PASSAGE DE 3 A 2 DIMENSIONS, PROBLEME DES UNITES ET REPRESENTATION DES AXES.

Considérons le premier système d'axes de référence que nous avons défini et prenons un point quelconque en 3 dimensions.
Soit P = (2, 3, 4).
Nous obtenons la représentation suivante :



Nous devrons être capables de déterminer la position physique du point P sur l'écran.

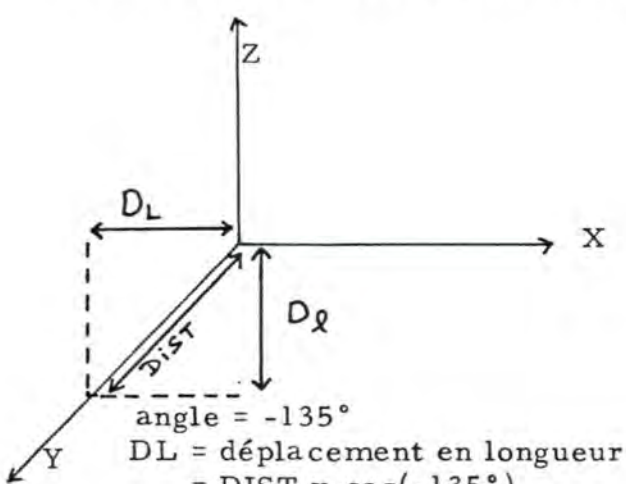


En 3 dimensions le point origine est (0, 0, 0). En deux dimensions il vaut (CX, CY) où CX et CY sont respectivement le milieu de la longueur et de la largeur de l'écran.

La première coordonnée d'un point logique à 3 dimensions cause un déplacement dans le sens de la longueur, la troisième dans le sens de la largeur. La deuxième coordonnée provoque un déplacement le long de l'axe debout, représenté par une oblique. Ainsi il y a un déplacement tant en longueur qu'en largeur.

Ce que nous voulons c'est reporter la distance sur cet axe. A cet effet nous déterminons les déplacements résultants en longueur et en largeur. Ils dépendent de la position de l'axe debout sur le schéma et leurs valeurs sont la distance à reporter sur l'axe multipliée respectivement par le cosinus et le sinus de l'angle formé par cet axe avec l'horizontale.

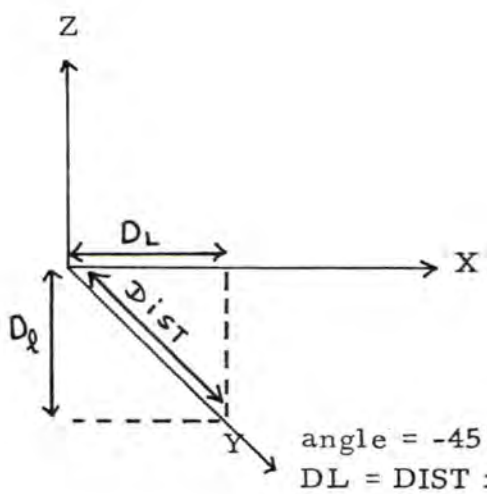
Voyons les deux exemples suivants :



angle = -135°

DL = déplacement en longueur
= DIST x cos(-135°)
= DIST x $(-\sqrt{2}/2)$

Dl = déplacement en largeur
= DIST x $(-\sqrt{2}/2)$



angle = -45°

DL = DIST x $\sqrt{2}/2$

Dl = DIST x $(-\sqrt{2}/2)$

Nous approchons la valeur $\sqrt{2}/2$ par la valeur 2/3, qui est un peu inférieure.

Nous définissons : MULX comme le facteur par lequel il faut multiplier la distance pour évaluer le déplacement dans le sens de la longueur (Ce sera donc une approximation de la valeur du cosinus).

MULY : idem dans le sens de la largeur (approximation de la valeur du sinus).

Nous utilisons des approximations, plutôt que les valeurs exactes des sinus et des cosinus pour 2 raisons :

- Ces valeurs approchées sont un peu inférieures aux valeurs réelles.
- Elles simplifient l'écriture des tests d'orientation. (voir points 4.2 et 4.3)

A partir de cela nous pouvons définir la procédure de passage de 3 à 2 dimensions (voir PASSER A 2 DIM).

- Le déplacement dans le sens de la longueur est égal à la somme de la première coordonnée et du produit de la 2e par MULX, le tout multiplié par l'unité.
- Le déplacement dans le sens de la largeur est égal à la somme de la 3e coordonnée et du produit de la 2e par MULY, le tout multiplié par l'unité.

Venons-en précisément à la manière de déterminer l'unité : il faut que chaque point (P1, P2, P3) que nous voulons représenter soit visible sur l'écran. Nous savons que la distance (P1+P2. MULX)* UNITE doit être inférieure au demi-axe en longueur et la distance (P3+P2 MULY) * UNITE doit être inférieure au demi-axe en largeur.

La procédure CHOIXUNITE renvoie une unité telle que les deux conditions soient satisfaites. Nous utilisons une constante MULTI. Nous avons fixé sa valeur à 0.8 pour que le dessin, tout en restant dans les limites de l'écran, soit suffisamment grand pour être représentatif. Elle sert de base à la détermination de l'unité en ce sens que ce choix est en général réalisé pour que l'un des 2 déplacements soit égal à 8/10 de la longueur du demi-axe correspondant.

Voyons les différents cas selon que P1, P2 et P3 sont ou non nuls.

-(0,0,0) : Ce point est toujours sur l'écran : il s'agit de (CX, CY). Nous prenons comme unité la longueur du demi-axe X (AXEX), fois 8/10.

-(0,0,P3) : Il s'agit d'assurer que ce point (situé sur l'axe X3) est visible : Nous prenons une unité telle que ce point soit situé à 8/10 de la hauteur du demi-axe Y (AXEY).

-(0, P2, 0) : il faut garantir que ce point sur l'axe X2 est visible; il faut donc que |(P2 * MULX * UNITE)| soit inférieur à AXEX et que |(P2 * MULY * UNITE)| soit plus petit que AXEY.

-(0, P2, P3) : Il faut que |(P2 * MULX * UNITE)| soit plus petit que AXEX; et |((P2 * MULY) + P3) * UNITE)| plus petit que AXEY.

-(P1, 0, 0) : Il faut que |(P1 * UNITE)| soit plus petit que AXEX.

-(P1, 0, P3) : Il faut que |(P1 * UNITE)| soit plus petit que AXEX. et |(P3 * UNITE)| plus petit que AXEY.

-(P1, P2, 0) : Il faut que |((P1 + P2*MULX) * UNITE)| soit plus petit que AXEX et que |(P2*MULY*UNITE)| soit plus petit que AXEY.

Remarque : Comme (P1 + P2*MULX) peut être nul, nous avons inversé l'ordre des tests.

-(P1, P2, P3) : C'est le cas général; rappelons qu'il faut :

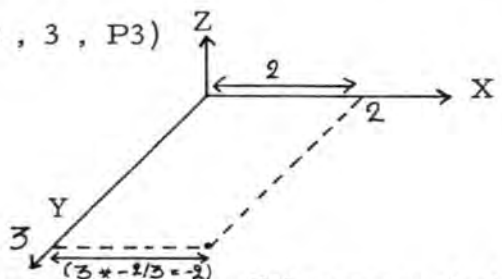
|(P1 + P2*MULX) * UNITE| plus petit que AXEX

et |(P3 + P2*MULY) * UNITE| plus petit que AXEY

De nouveau il faut prendre garde à diviser par 0 ce qui pourrait arriver si (P1 + P2*MULX) ou (P3 + P2*MULY) sont nuls.

Si P1 + P2*MULX = 0 cela signifie que le déplacement en longueur est nul.

Exemple (2 , 3 , P3)

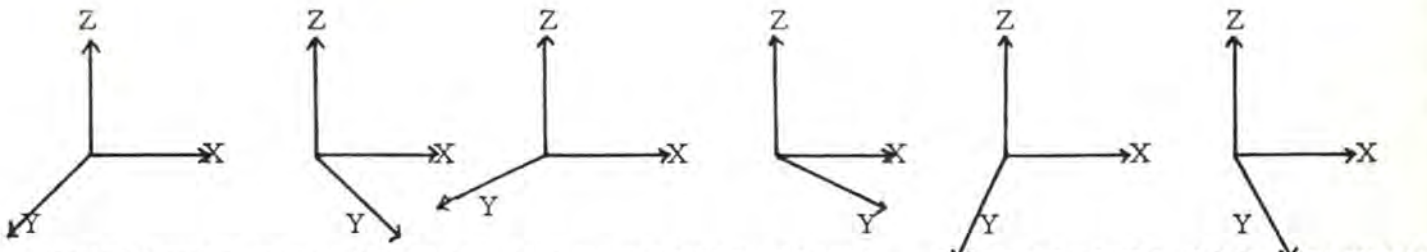


Ce point a donc CX comme abcisse et l'unité doit être prise suffisament élevée (1000) pour que l'on la recalcule par rapport à P3.

4.2. PROBLEME DES ORIENTATIONS ET REPRESENTATION DES AXES.

Nous savons que plusieurs systèmes de références sont nécessaires pour représenter les plans.

Nous en avons déjà présenté 3 : il se peut que ce ne soit pas suffisant pour superposer 2 ou 3 plans sur un même schéma car si le premier plan n'est pas visible dans les deux premiers systèmes de référence et que le deuxième ne l'est pas dans le troisième, nous sommes bloqués. Il faut donc définir plus de systèmes; nous en avons choisi 6. Les valeurs MULX et MULY permettent de situer les points sur cet axe.



ORIENTATION NR0 1	ORIENTATION NR0 2	ORIENTATION NR0 3	ORIENTATION NR0 4	ORIENTATION NR0 5	ORIENTATION NR0 6
MULX = -2/3	MULX = 2/3	MULX=-2/3	MULX =2/3	MULX = -1/3	MULX = 1/3
MULY = -2/3	MULY = -2/3	MULY=-1/3	MULY=-1/3	MULY = -2/3	MULY = -2/3

Nous avons défini une matrice ORI, qui pour les 6 orientations possibles contient à chaque fois la valeur du MULX et du MULY correspondants.
La procédure qui trace les axes (voir TRAXES) dépend elle aussi de MULX et de MULY puisque la position de l'axe X2 en dépend. De plus nous avons vu dans les exemples que nous ne représentons la partie négative de l'axe que si c'était nécessaire, c.a.d. si une valeur négative sur un axe nous sert de point de repère. Pour cela nous devrons positionner les indicateurs. AXE1NEG, AXE2NEG, et AXE3NEG en choisissant nos sommets.
Nous traçons l'axe X1 de (XMAX, CY) à (0, CY) ou (CX, CY) selon qu'il faut ou non représenter sa partie négative.
Nous traçons de même l'axe X3 de (CX, YMAX) à (CX, 0) ou (CX, CY).
Pour l'axe X2 c'est un peu plus compliqué puisque sa position peut changer ; selon la pente de l'axe le point-limite avec l'écran peut se situer sur les bords gauche, droit ou inférieur. Dans les 2 premiers cas la longueur du demi-axe est égal à |(96/MULY)| et dans le troisième cas à |(140/MULX)|.
Il suffit alors de chercher la coordonnée du point-limite et selon qu'il faut représenter la partie négative ou non, prendre comme deuxième point soit le point opposé, soit le centre.

4.3. CHOIX DES POINTS ET DU SYSTEME DE REFERENCE POUR REPRESENTER UN PLAN.

(Voir procédure QUELPLAN).
Nous distinguons 16 types de plans, selon que l'un ou l'autre des 4 coëfficients A, B, C, D est ou non nul.
Les 16 cas sont les suivants : (à chaque cas nous avons associé une procédure de nom PLI, ou I est le numéro du cas. Cette procédure détermine les sommets du parallèlogramme et les orientations qui ne sont pas permises pour ce type de plan. Ainsi lorsque nous devrons superposer 2 plans nous saurons quelles orientations éviter).

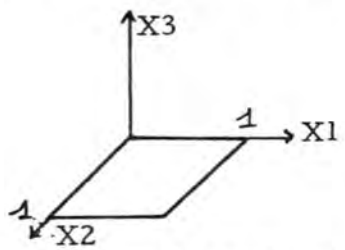
Nom de la procédure.	Type d'équation examinée.
PL1	0 = 0 (indéterminé)
PL2	CZ = 0 (plan horizontal)
PL3	BY = 0 (plan de face)
PL4	BY + CZ = 0
PL5	AX = 0 (plan debout)
PL6	AX + CZ = 0
PL7	AX + BY = 0
PL8	AX + BY + CZ = 0
PL9	0 = D (impossible)
PL10	CZ = D (// au plan horizontal)
PL11	BY = D (// au plan de face)
PL12	BY + CZ = D
PL13	AX = D (// au plan debout)
PL14	AX + CZ = D
PL15	AX + BY = D
PL16	AX + BY + CZ = D

Examinons succesivement les 16 procédures :

PL1 : Il n'y a pas de plan à représenter. Le nombre de sommets à chercher est égal à 0.

PL2 : Il faut représenter le plan horizontal. Nous définissons 4 sommets : (0, 0, 0) (1, 0, 0) (1, 1, 0) (0, 1, 0).

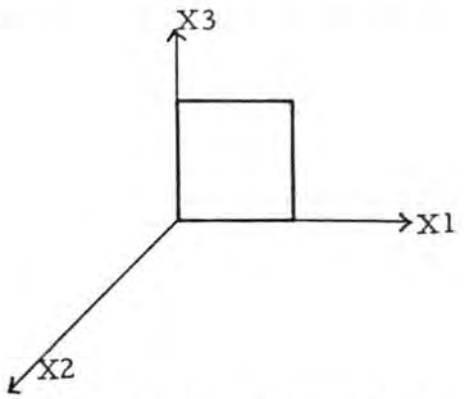
Nous choisissons le premier système de référence : MULX = -2/3 et MULY = - 2/3.



Nous devons également déterminer si certaines orientations ne sont pas "interdites" pour ce type de plan. Il apparait que dans ce cas, les 6 systèmes de référence définis conviennent tous.

PL3: Nous représentons le plan de face grâce aux 4 sommets suivants : (1,0,0) (0,0,0,) (0,0,1) (1,0,1).

De même nous choisissons le premier système de référence et aucun autre n'est interdit.

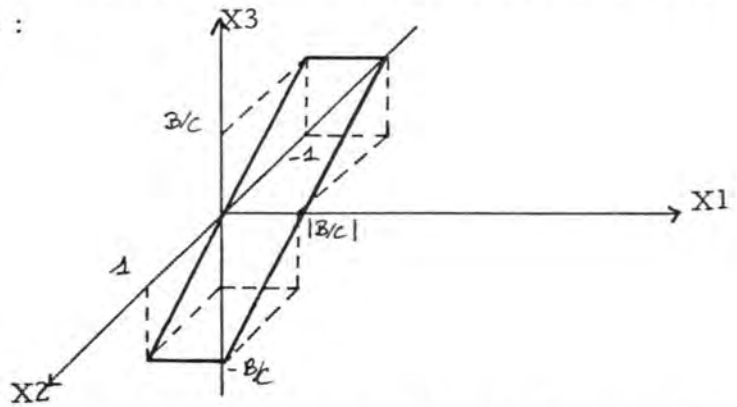


PL4: Il s'agit d'un plan passant par l'origine et parallèle à l'axe X1. La droite dans le plan debout passant par (0, 0, 0) est définie par les 2 points (0, 1, -B/C) et (O, -1, B/C).

Il faut encore définir 2 autres sommets le long de l'axe X1. Pour obtenir un schéma homogène (c.a.d. où tous les côtés sont dans le même ordre de grandeur) nous prenons comme autres points (|B/C|, 1, -B/C) et (|B/C|, -1, B/C).

Nous choisissons le premier système de référence pour le représenter.

Nous obtenons :

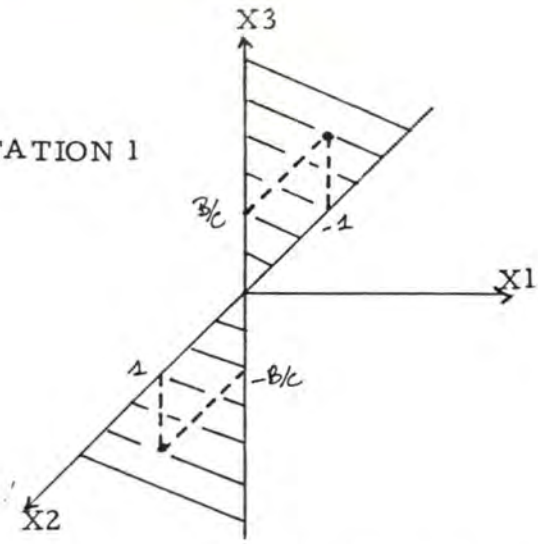


Par contre dans ce cas il peut survenir des problèmes d'orientation, si la droite passant par (0, 1, -B/C) et (0, -1, B/C) est proche d'une horizontale, nous ne verrons plus rien. Il faut donc déterminer quels sont les cas où cela peut arriver. C'est le but de la procédure TESTOR1 (Premier test d'orientation). S'il y a problème elle fixe MULX et MULY (-2/3, -1/3) pour permettre la représentation du plan.

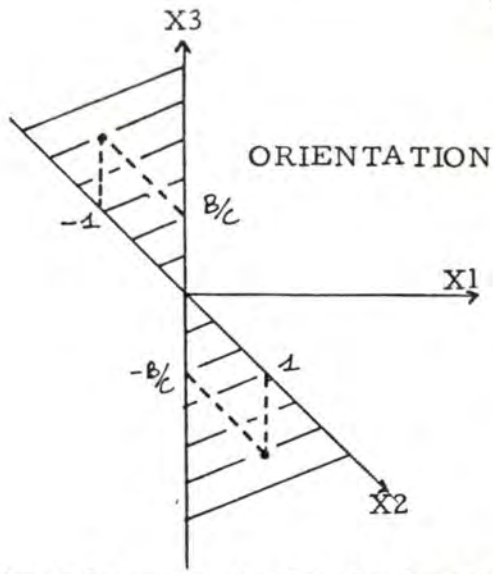
Nous pourrions vérifier sur les 6 schémas que si B et C sont du même signe (ou B * C plus grand que 0), il n'y aura pas de problème car la droite ne peut être horizontale.

Voyons le sur les 2 premiers schémas :

ORIENTATION 1



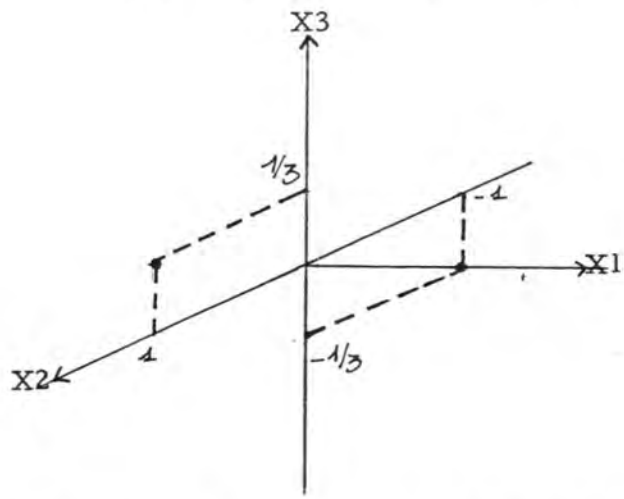
ORIENTATION 2

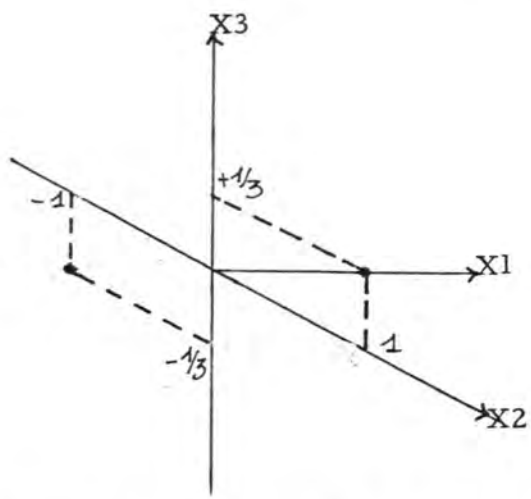


Quels que soient B et C la droite sera dans la partie hachurée du schéma donc elle ne pourra être horizontale. Par contre si B et C sont de signes opposés il faudra veiller à ce que le rapport $|B/C|$ ne soit pas trop proche de $|MULY|$, car alors cette droite serait plus ou moins horizontale, puisque le déplacement dans le sens de la largeur (qui est P2*MULY + P3) serait plus ou moins nul.

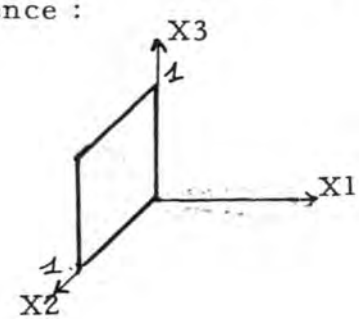
Voyons aussi 2 exemples :

Si B x C 0 et (B/C) 1/3 les orientations 3 et 4 sont à rejeter.





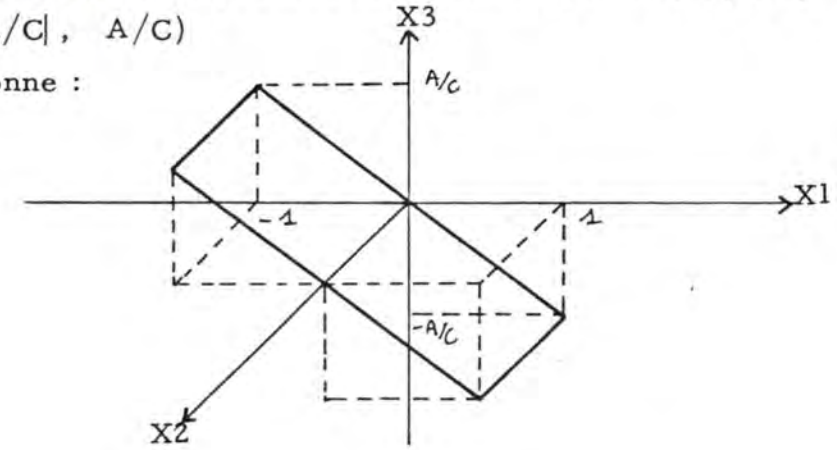
PL5: Pour représenter le plan debout , nous prenons les 4 sommets suivants: (0, 0, 0,) (0, 1, 0) (0, 1, 1,) (0, 0, 1) ce qui donne dans le premier système de référence :



Cette représentation est valable pour les 6 orientations.

PL6: Il s'agit d'un plan passant par l'origine et parallèle à l'axe X2. Nous définissons la trace de ce plan sur le plan de face par les points (1, 0, -A/C) et (-1, 0, A/C). Nous prenons les 2 autres points (1, |A/C|, -A/C) et (-1, |A/C|, A/C)

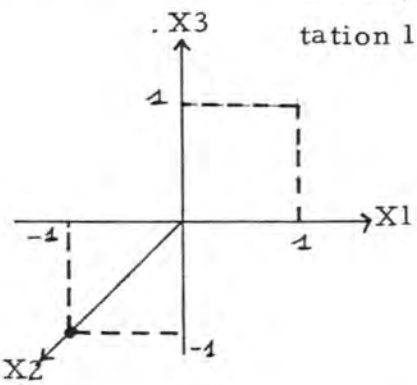
cela donne :



Il peut y avoir des problèmes de représentation si la pente de la droite d'intersection avec le plan de face est la même que celle de l'axe X2. La procédure TESTOR2 vérifie si le rapport entre A et C n'est pas le même que celui entre MULX et MULY, au signe opposé.

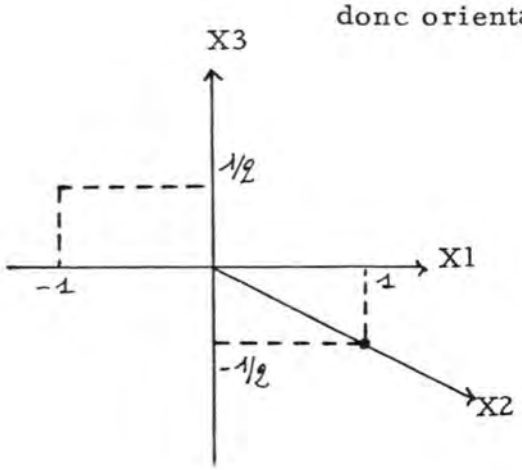
Voyons deux exemples : X - Z = 0 (1, 0, 1) (-1, 0, -1) A/C = -1

MULX/MULY ne peut être = +1, donc orientation 1 interdite.

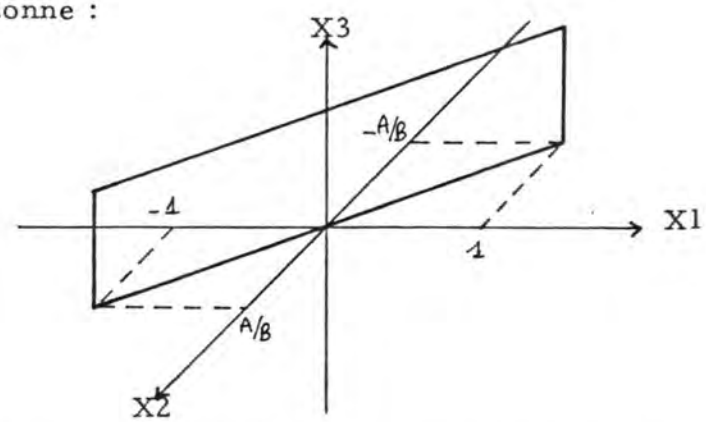


X + 2Z = 0 (1, 0, -1/2) (-1, 0, 1/2) A/C = 1/2

MULX/MULY doit être différent de -1/2, donc orientation 6 interdite.



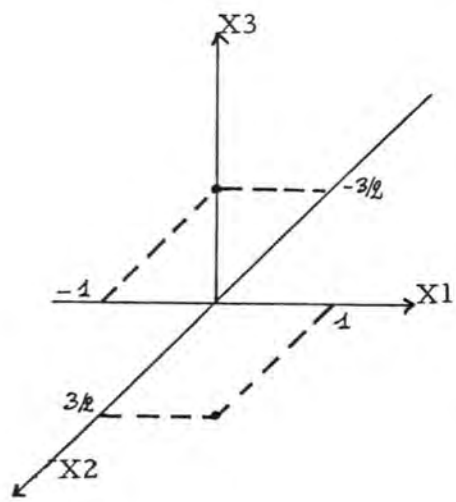
PL7 : Le plan passe par l'origine et est parallèle à l'axe X3.
Nous définissons la trace sur le plan horizontal grâce aux 2 points (1, -A/B, 0) et (-1, A/B, 0).
Nous choisissons |A/B| comme hauteur pour les 2 autres points.
Cela donne :

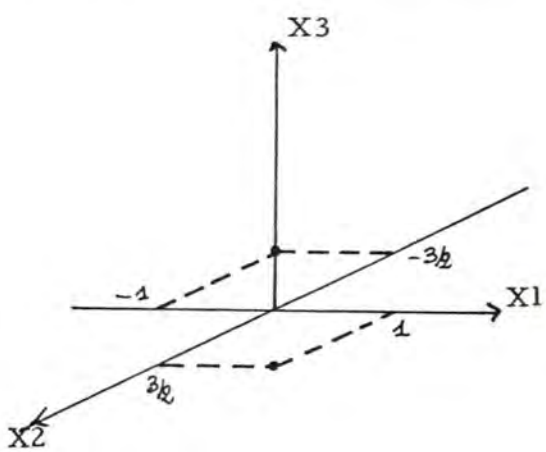


Il peut y avoir des problèmes d'orientation si la droite d'intersection avec le plan horizontal a une pente proche de la verticale.
Pour les 3 orientations où l'axe X2 est situé à gauche de la verticale (1,3,5) cela surviendra si A et B sont de signes opposés et si $|A/B.\ MULX| \simeq |1|$, c. a. d. si $|B/A| \simeq MULX$.

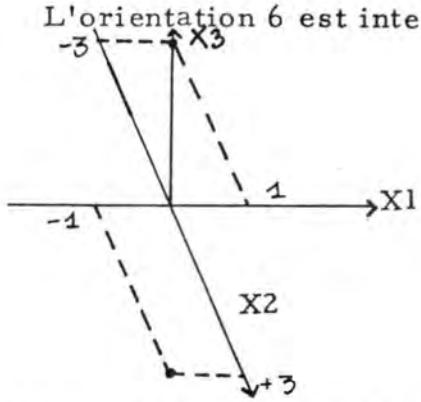
Pour les 3 autres orientations (2,4,6) cela surviendra dans le même cas si A et B sont de même signe.

Voyons 2 exemples : $3X - 2Y = 0$ (1, 3/2, 0) (-1, -3/2, 0)
$B \times A < 0$; $|B/A| = |2/3|$
MULX doit être différent de -2/3.
Les orientations 1 et 3 sont interdites.



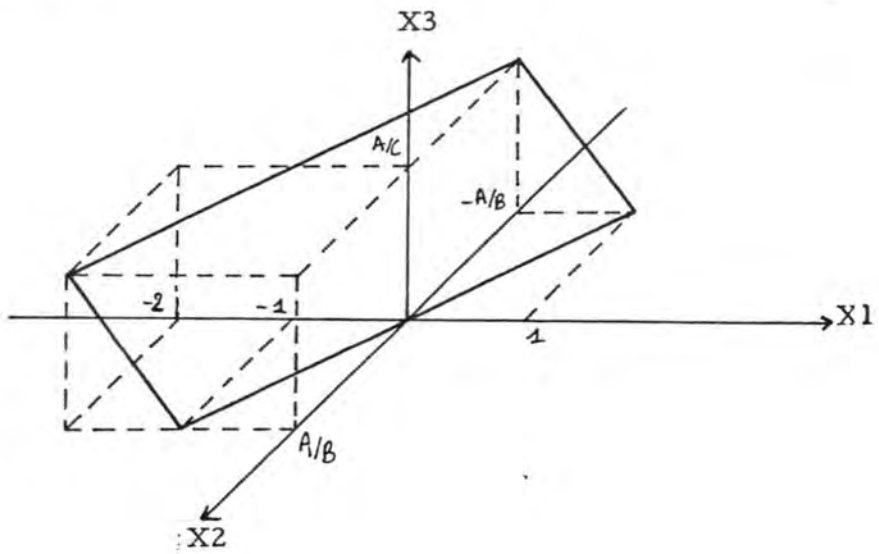


$3X + Y = 0$. $(1, -3, 0)$ $(-1, 3, 0)$
$B \times A > 0$. $|B/A| = 1/3$
MULX doit être différent de $1/3$.
L'orientation 6 est interdite.



En plus TESTOR3 fixe également MULX et MULY (2/3, -2/3) en cas de problème pour permettre une représentation correcte.

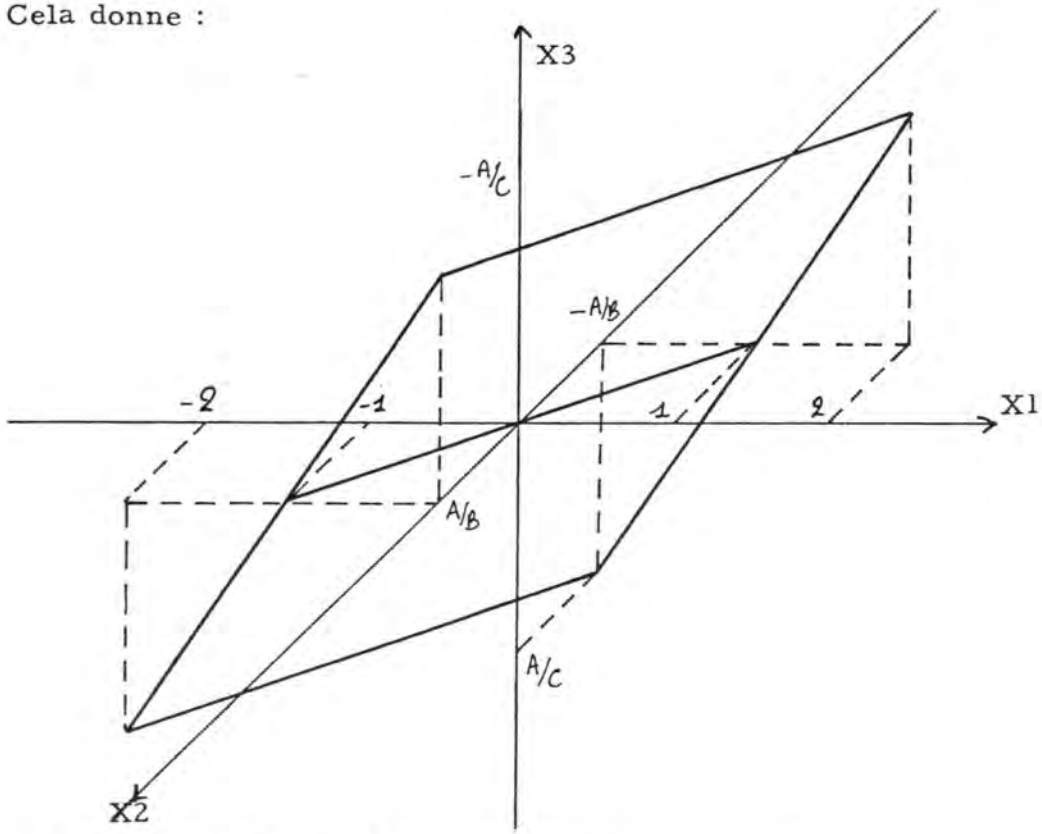
PL8 : Il s'agit du type de plan où seul le terme indépendant D est nul. Nous avons vu que pour le représenter nous utilisons les traces du plan sur les plans horizontal et de face.
Nous choisissons les 2 points $(1, -A/B, 0)$ et $(-1, A/B, 0)$ pour représenter le premier segment. Comme nous connaissons l'équation de la droite d'intersection avec le plan de face ($AX + CZ = 0$; $Y=0$), nous pouvons définir les 2 autres points : $(-2, A/B, A/C)$ et $(O, -A/B, A/C)$. Nous obtenons les 2 points en partant des 2 précédents; lorsque nous nous déplaçons d'une unité sur l'axe X1, nous nous déplaçons de $(-A/C)$ sur l'axe X3.
Cela donne :



Comme en plus nous avons décidé de "doubler" la figure si la partie représentée du plan est sous le niveau de l'horizontale, nous serons amenés à définir 2 sommets supplémentaires qui sont (0, A/B, -A/C) et (2, -A/B, -A/C).

Nous faisons cela si A * C < 0 .

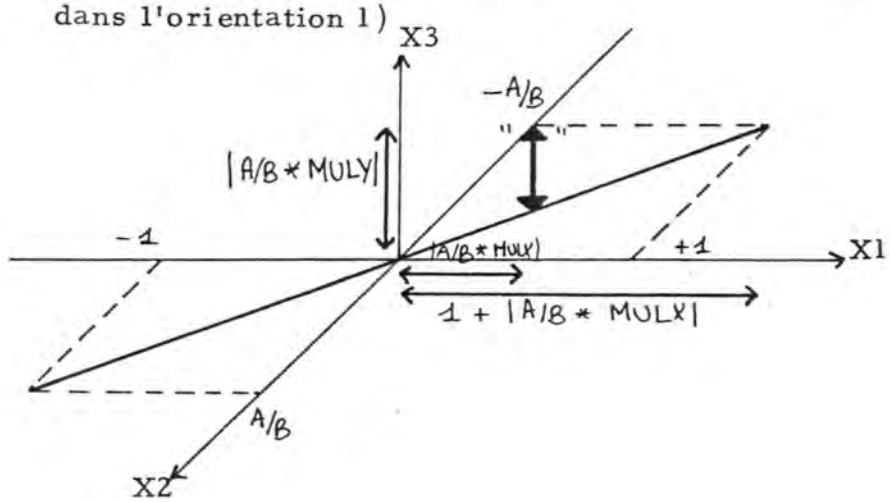
Cela donne :



Nous positionnons alors le nombre de sommets à 6.

Nous nous rendons compte que si les 2 droites sont représentées dans un système de référence tel que leurs pentes sur l'écran (donc en fonction de MULX et MULY) sont proches, nous ne verrons plus rien.

Ce fait est assez complexe à détecter: il faut distinguer plusieurs cas.: - Si A * B est plus grand que 0 et que nous sommes dans un système de référence où l'axe X2 est situé du côté gauche de la verticale (orientations 1,3,5) nous avons : (par ex. dans l'orientation 1)



Nous savons que le troisième sommet est obtenu en élevant la distance A/C à partir de (0, -A/B, 0).

Si A/C est plus petit que 0 et $|A/C|$ = "↕", ce sommet est situé sur l'autre droite et nous ne voyons plus rien.

Que vaut cette distance "↕"?

D'après la pente de la droite, lorsque nous "avançons" de $1 + (|A/B * MULX|)$, nous "montons" de $(|A/B.\ MULY|)$.

Ainsi si nous "avançons" de A/B. MULX nous "montons" de $|A/B MULY| * \frac{|A/B \times MULX|}{1 + |A/B \times MULX|}$.

Cette expression représente la différence entre le distance (AB x MULY) et la distance "↕".

Ainsi si cette expression est proche de $|A/B * MULY| - |A/C|$, il y aura problème.

Nous avons donc :

$$\left|(A/B * |MULY|) - |A/C|\right| \not\approx \left|(A/B * |MULY|)\frac{|A/B * MULX|}{1 + |A/B * MULX|}\right|$$

ou encore

$$\left|(A/B * |MULY|)\left(1 - \frac{|A/B * MULX|}{1 + |A/B * MULX|}\right)\right| \not\approx |A/C|$$

Pour l'orientation 1 : MULX = -2/3 et MULY = - 2/3 nous pouvons écrire $\frac{2A}{3B}\left(1 - \frac{2A/3B}{1 + 2A/3B}\right) \not\approx |A/C|$

$$\frac{2A}{3B}\left(1 - \frac{2A}{3B+2A}\right) \not\approx |A/C|$$

$$\frac{2A}{3B}\left(\frac{3B + 2A - 2A}{3B + 2A}\right) \not\approx |A/C|$$

$$\frac{2A}{3B + 2A} \not\approx |A/C|$$

Pour l'orientation 3 : MULX = -2/3 et MULY -1/3.

nous avons : $\frac{A}{3B + 2A} \not\approx |A/C|$

Pour l'orientation 5 : MULX = - 2/3 et MULY - 2/3.

nous avons : $\frac{2A}{3B + A} \not\approx |A/C|$

Tous ces tests (ainsi que ceux qui seront décrits ci-dessous) sont effectués par la procédure TESTOR4.

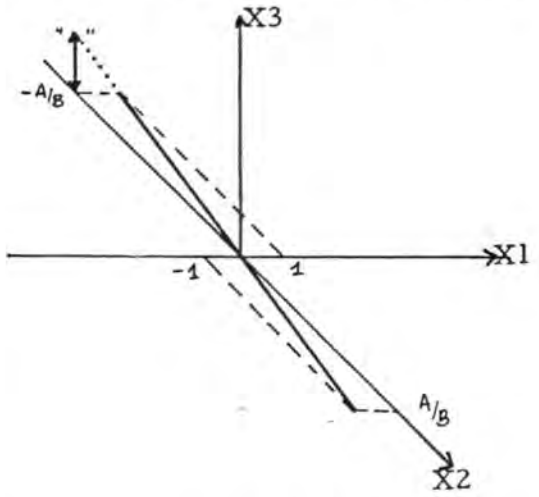
Pour réaliser ces tests nous avons recours à la fonction APEUPRES. Celle-ci a 2 paramètres qui sont 2 nombres dont elle compare la valeur absolue. Si ces deux nombres sont "plus ou moins" égaux, la fonction est vraie, sinon elle est fausse.

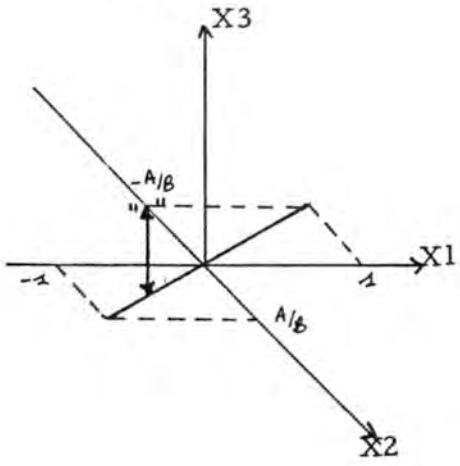
La marge de différence acceptée entre les 2 nombres est FPP (facteur à peu près). Nous l'avons fixé à 0.8.

Si la première orientation est interdite, nous affectons une valeur à MULX (2/3) et à MULY (-2/3) pour permettre une bonne repré-

sentation.

Considérons toujours A * B plus grand que 0, mais prenons les 3 autres orientations (2, 4, 6). Par exemple l'orientation 2 :





Dans ce cas $|A/B*MULX|>1$ et il y a un problème si A/C est >0 et si $|A/C| \simeq$ distance "↕".

Nous pouvons évaluer cette distance comme tout à l'heure et

$$\left|\frac{A}{C}\right| \neq \left|\left|\frac{A}{B}MULY\right|\left(\frac{|A/B*MULX|}{|A/B*MULX|-1} - 1\right)\right|$$

Dans ce cas $|A/B*MULX|<1$ et il y a un problème si A/C est <0 et si $|A/C| \simeq$ distance "↕".

Nous obtenons ici également

$$\left|\frac{A}{C}\right| \neq \left|\left|\frac{A}{B}MULY\right|\left(\frac{|A/B*MULX|}{1-|A/B*MULX|} + 1\right)\right|$$

Ce qui donne pour l'orientation 2 : MULX = 2/3 , MULY =-2/3

$$\left|\frac{A}{C}\right| \neq \left(\frac{|2\,A|}{|3B| - |2A|}\right) \qquad \left|\frac{A}{C}\right| \neq \frac{|2\,A|}{(|3B| - |2A|)}$$

Pour l'orientation 4 : MULX = 2/3 , MULY = -1/3

$$\left|\frac{A}{C}\right| \neq \left(\frac{|A|}{|3B| - |2A|}\right) \qquad \left|\frac{A}{C}\right| \neq \frac{|A|}{(|3B| - |2A|)}$$

Pour l'orientation 6 : MULX = 1/3 et MULY -2/3.

$$\left|\frac{A}{C}\right| \neq \left(\frac{|2\,A|}{|3B| - |A|}\right) \qquad \left|\frac{A}{C}\right| \neq \left(\frac{|2\,A|}{|3B| - |A|}\right)$$

Avec le même genre de raisonnement, nous obtenons les contraintes suivantes si A * B < 0.

Pour l'orientation 1 :

$$\text{Si } \left|\frac{A}{B} * MULX\right| > 1 \text{ alors } \frac{A}{C} \neq \left|\frac{|2A|}{|2A - 3B|}\right|$$

$$\text{Si } \left|\frac{A}{B} * MULX\right| < 1 \text{ alors } \frac{A}{C} \neq \left|\frac{|2A|}{|3B - 2A|}\right|$$

Pour l'orientation 3 :

$$\text{Si } \left|\frac{A}{B} * MULX\right| > 1 \text{ alors } \frac{A}{C} \neq \left|\frac{|A|}{|2A - 3B|}\right|$$

$$\text{Si } \left|\frac{A}{B} * MULX\right| < 1 \text{ alors } \frac{A}{C} \neq \left|\frac{|A|}{|3B - 2A|}\right|$$

Pour l'orientation 5 :

$$\text{Si}\left|\frac{A}{B} * \text{MULX}\right| > 1 \text{ alors } \left|\frac{A}{C}\right| \neq \frac{2A}{A - 3B}$$

$$\text{Si}\left|\frac{A}{B} * \text{MULX}\right| < 1 \text{ alors } \left|\frac{A}{C}\right| \neq \frac{2A}{3B - A}$$

Pour l'orientation 2 :

$$\left|\frac{A}{C}\right| \neq \frac{|2A|}{|3B + |2A|}$$

Pour l'orientation 4 :

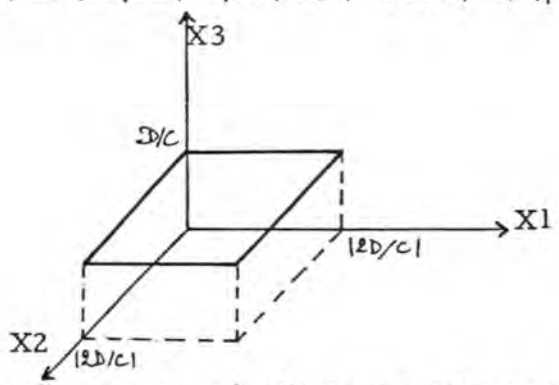
$$\left|\frac{A}{C}\right| \neq \frac{|A|}{|3B + |2A|}$$

Pour l'orientation 6 :

$$\left|\frac{A}{C}\right| \neq \frac{|2A|}{|3B + |A|}$$

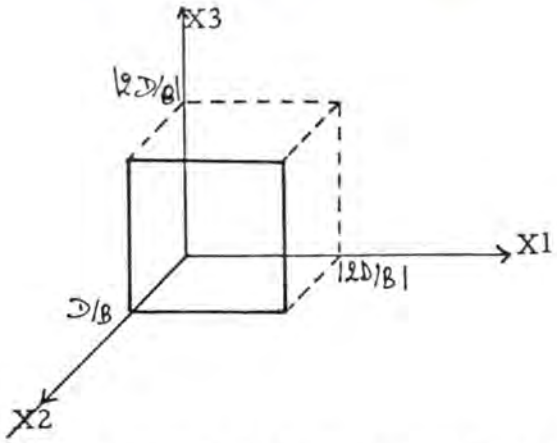
PL9: Il s'agit de l'équation impossible: comme iln'y a rien à représenter le nombre de sommets = 0.

PL10: Nous devons représenter un plan parallèle au plan horizontal.
Nous choisissons les 4 points suivants:
$(|2D/C|, |2D/C|, D/C)$ $(0, |2D/C|, D/C)$ $(0, 0, D/C)$ $(|2D/C|, 0, D/C)$
Ce qui donne :



Si C et D sont des signes opposés l'intersection avec l'axe X3 sera située sous le niveau de l'horizontale; dans ce cas nous positionnons AX3NEG à vrai de manière à représenter aussi la partie négative de cet axe.
Il n'y aura pas de problème de représentation pour ce type de plan.

PL11: Il s'agit d'un plan parallèle au plan de face. Nous avons choisi les 4 points suivants :
$(|2D/B|, D/B, 0)$ $(0, D/B, 0)$ $(0, D/B, |2D/B|)$ $(|2D/B|, D/B, |2D/B|)$.



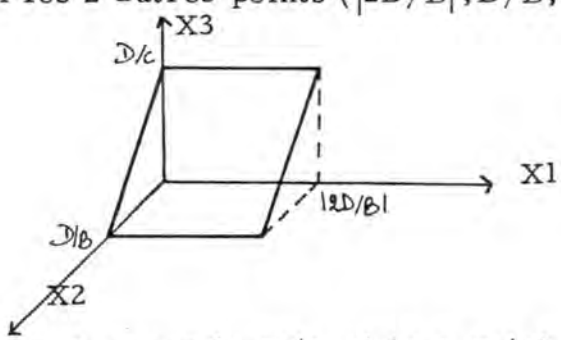
Si D * B < 0 nous positionnons AX2NEG pour représenter la partie négative de cet axe.

Il n'y aura pas de problème d'orientation.

PL12: Il s'agit d'un plan parallèle à l'axe X1.

Nous représentons la droite d'intersection avec le plan debout grâce aux deux points (0,D/B,0) (0,0,D/C).

Nous avons choisi les 2 autres points (|2D/B|,D/B,0) (|2D/B|,0,D/C).



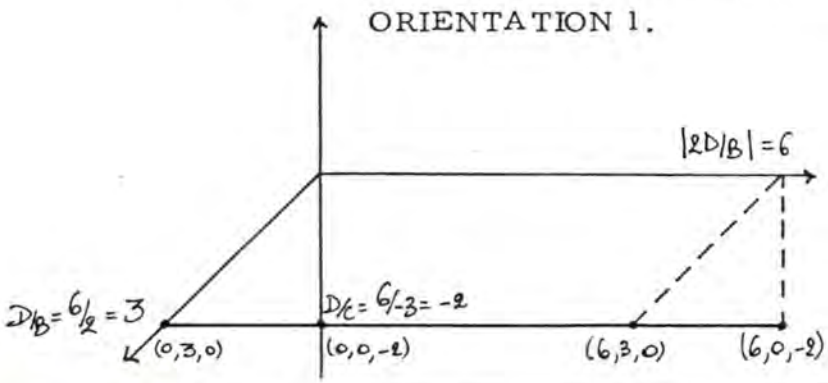
Si l'un ou l'autre des 2 points (0,D/B,0) (0,0,D/C) se situe dans la partie négative de l'axe correspondant, nous positionnons l'indicateur de manière à représenter cette partie de l'axe aussi.

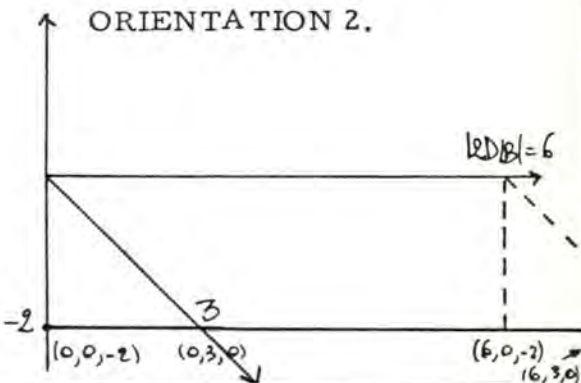
Nous pouvons avoir des problèmes d'orientation : si la droite d'intersection avec le plan debout se représente proche d'une horizontale nous ne verrons plus rien (voir PL4 et TESTOR1).

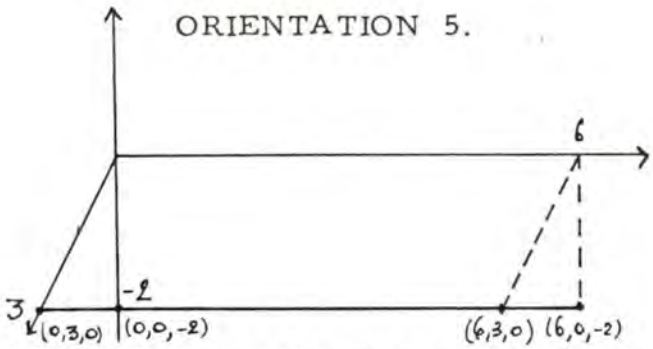
Nous pouvons reprendre la même discussion: si B et C sont de même signe il n'y a pas de problème. Par contre s'ils sont de signe opposé et que |B/C| ≠ |MULY| il y a problème.

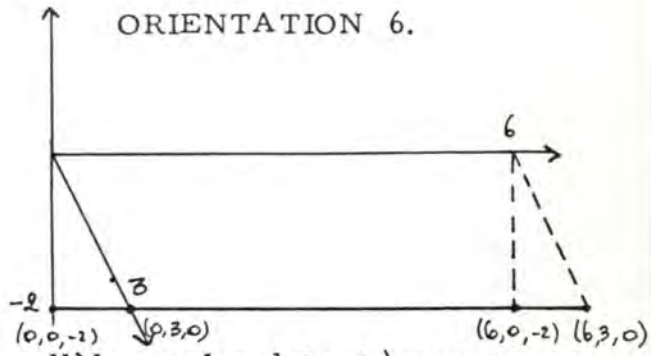
Voyons 4 exemples différents de tout à l'heure :

2Y - 3C = 6 ⟶ les orientations 1,2,5,6 sont interdites.

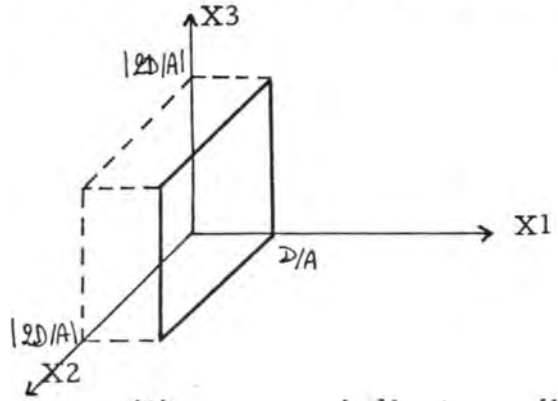






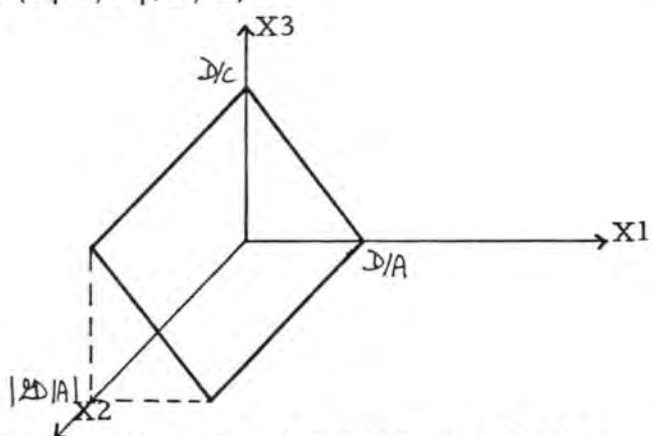


PL13: Pour représenter ce type de plan (parallèle au plan debout) nous avons choisi les 4 points suivants :
(D/A, 0, 0) (D/A, |2D/A|, 0) (D/A, |2D/A|, |2D/A|) (D/A, 0, |2D/A|).



Comme d'habitude nous positionnons un indicateur s'il est nécessaire de représenter la partie négative de l'axe.
Il n'y aura pas de problème d'orientation .

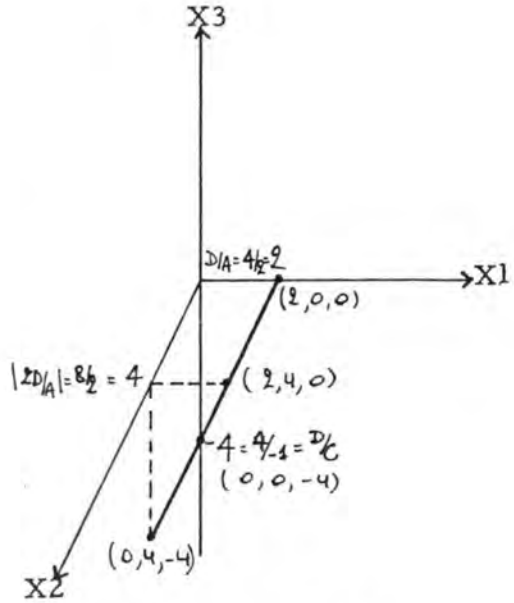
PL14: Nous devons représenter ce plan parallèle à l'axe X2.
Nous représentons la trace du plan sur le plan de face (points(D/A, 0, 0) (0, 0, D/C)). Nous prolongeons alors parallèlement à X2. Les points sont (D/A, |2D/A|, 0,) (0, |2D/A|, D/C)



Si nécessaire, nous positionnons AX1NEG et AX3NEG. Les problèmes d'orientation sont les mêmes que pour PL6; il ne faut pas que le segment (D/A, 0, 0) - (0, 0, D/C) ait la même pente que l'axe X2 (voir TESTOR2).

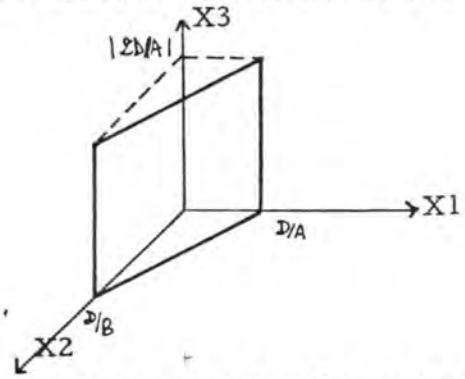
Voyons un exemple :

2X - Z = 4 : comme A * C < 0 et $|A/2| \neq |C|$, l'orientation 5 est interdite.



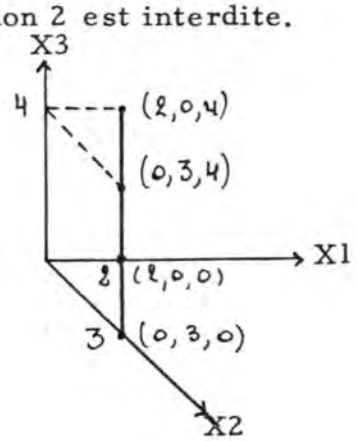
PL15: Il s'agit cette fois d'un plan parallèle à X3.

Nous représentons la trace du plan sur le plan horizontal. Les 4 points sont : (D/A, 0, 0) (0, D/B, 0) (0, D/B, |2D/A|) (D/A, 0, |2D/A|)

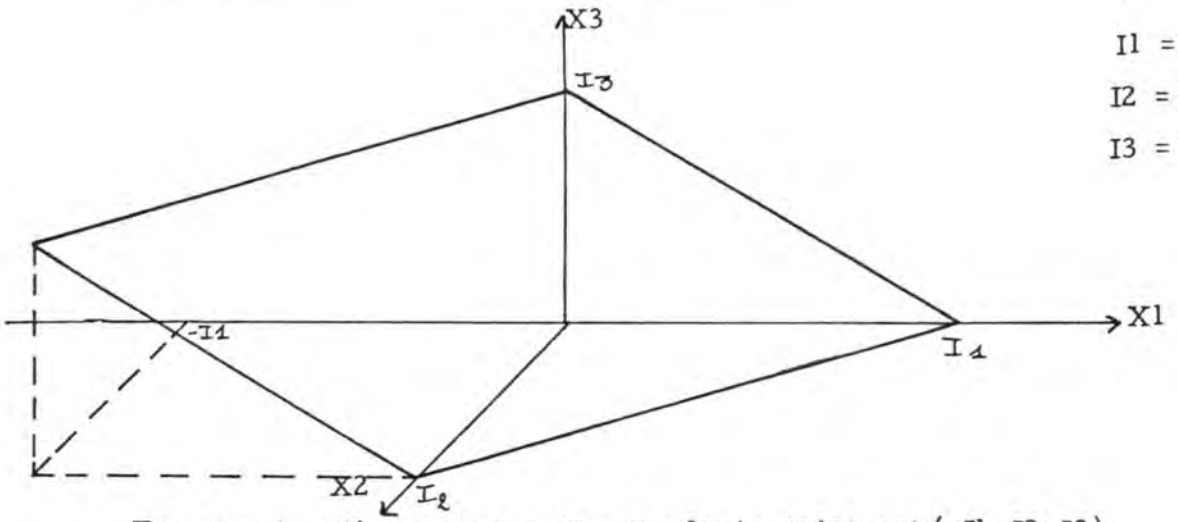


Si nécessaire nous positionnons AX1NEG et AX2NEG. Les problèmes d'orientation sont les mêmes que pour PL7: il ne faut pas que la représentation de la droite sur le plan horizontal apparaisse proche de la verticale. (voir TESTOR 3).

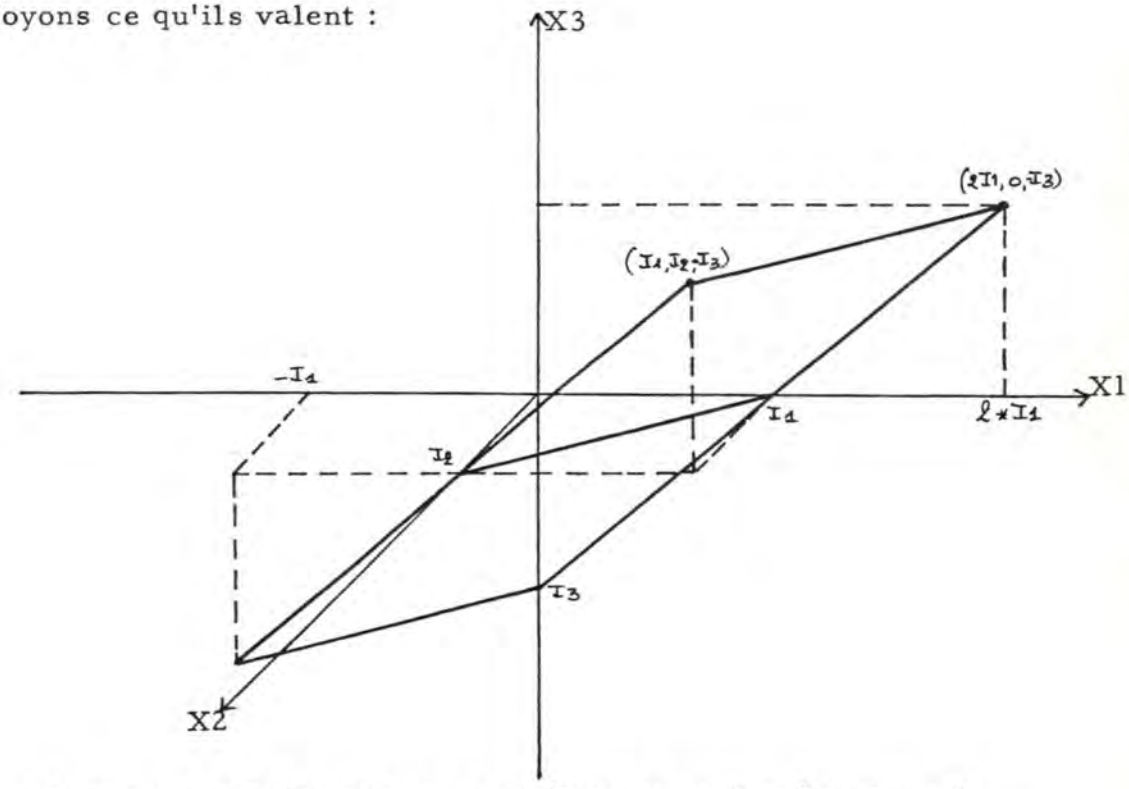
Voyons l'exemple suivant : 3X + 2Y = 6. Comme A * B < 0 et que $|B/A| \simeq |2/3|$, l'orientation 2 est interdite.



PL16: Nous avons vu que nous représentons ce type de plan à partir de ses 3 points d'intersection avec les axes.



Par construction nous voyons que le 4e point est (-I1, I2, I3).
Les 4 points sont donc : (I1, 0, 0) (0, I2, 0) (-I1, I2, I3) (0, 0, I3).
Nous avons pris comme convention de "doubler" la figure si elle se situe sous le niveau horizontal, c.a.d. si I3 < 0. Dans ce cas nous définissons 2 autres sommets (NBSOMMET = 6).
Voyons ce qu'ils valent :

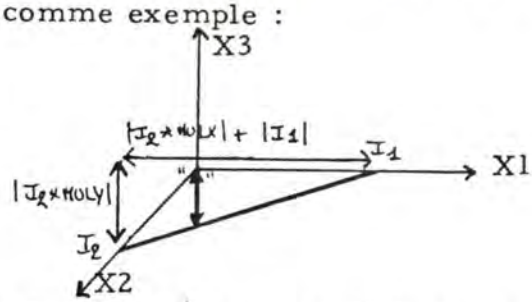


Les 2 points supplémentaires sont (2*I1, 0, -I3) et (I1, I2, -I3).
Si l'un ou l'autre des points I1, I2 ou I3 est négatif nous positionnons bien sûr les indicateurs correspondants pour dessiner les parties négatives de ces axes.
Les problèmes d'orientation sont du même genre que pour PL8: il ne faut pas que les représentations des 2 droites soient plus ou moins de même pente.

Reprenons la même découpe que pour PL8 (voir TESTOR4)

- Si $A * B > 0$ considérons les orientations 1, 3, 5.

Prenons la 1ère comme exemple :



Si I3 est négatif et que $|I3| \simeq$ "↕", le parallèlogramme sera complètement "raplati".

Nous pouvons tenir le même raisonnement :

en($|I1| + |MULX * I2|$) nous "descendons" de $|MULY * I2|$

donc en $|I1|$ nous descendons de $|MULY * X2| \dfrac{|I1|}{|I1| + |MULX * I2|}$

et nous obtenons la condition suivante :

$$|I3| \not\simeq \frac{|I1 * I2 * MULY|}{|I1| + |MULX * I2|}$$

Cette expression n'est pas satisfaisante, car nous y avons inclus le terme indépendant "D" alors que celui-ci n'a pas d'influence sur la pente des droites. Essayons de l'éliminer.

Remplaçons I1, I2 et I3 par leurs valeurs :

$$\left|\frac{D}{C}\right| \not\simeq \frac{|D/A * D/B * MULY|}{|D/A| + |D/B * MULY|}$$

$$\left|\frac{1}{C}\right| \not\simeq \frac{|1/A * D/B * MULY|}{|D|(|1/A| + |1/B * MULY|)}$$

$$\left|\frac{1}{C}\right| \not\simeq \frac{|MULY / AB|}{(|B| + |A * MULY|)/AB}$$

$$\left|\frac{1}{C}\right| \not\simeq \frac{|MULY|}{|B| + |A * MULX|}$$

Voyons ce que cela donne pour les différentes orientations :

1 : MULX = -2/3 MULY = -2/3

$$\left|\frac{1}{C}\right| \not\simeq \frac{2/3}{|B| + 2/3\,A|} \quad \text{ou} \quad \left|\frac{1}{C}\right| \not\simeq \frac{2}{|3B| + |2A|}$$

Si nous multiplions les 2 membres par A nous obtenons

$\left|\dfrac{A}{C}\right| \not\simeq \dfrac{|2A|}{|3B| + |2A|}$, ce qui est exactement la condition que nous avions trouvée pour PL8 dans le même cas.

3 : MULX = -2/3 MULY = -1/3

$$\left|\frac{A}{C}\right| \not\simeq \frac{|A|}{|3B| + |2A|}$$

5 : MULX = -1/3 MULY = -2/3

$$\left|\frac{A}{C}\right| \not\simeq \frac{|2A|}{|3B| + |A|}$$

Remarquons que dans TESTOR4 nous n'utilisons pas les valeurs absolues car, d'une part A et B sont de même signe et d'autre part la fonction APEUPRES compare les valeurs absolues des nombres.

Si A et B > 0 $\quad 3B + 2A$ est équivalent à $|3B| + |2A|$

Si A et B < 0 $\quad 3B + 2A$ est l'opposé de $|3B| + |2A|$

mais comme APEUPRES traite la valeur absolue il n'y a pas de problème.

Si nous reprenons le même type de raisonnement nous verrons que la procédure TESTOR4 (voir PL8) est entièrement d'application ici.

4.4. SUPERPOSITION DE 2 PLANS.

4.4.1. Rappel de la méthode suivie.

Au point 2.2.2 nous avons expliqué la manière dont nous envisageons résoudre ce problème.

Pour rappel, l'idée de départ était de représenter l'intersection entre 2 plans par le segment qui lie le point d'intersection entre les 2 plans sur le plan horizontal et celui sur le plan de face. Nous avons vu que dans certains cas (1 ou 2 variables simultanément absentes des 2 équations) l'un ou l'autre de ces points pouvaient ne pas exister.

C'est pour cela que nous devons traiter ces différents cas d'une autre façon.

Nous verrons au point 4.4.3 la découpe qui en résulte.

Nous avons vu également que souvent il est nécessaire de "prolonger" un plan afin de permettre de situer un point d'intersection. C'est ce que nous allons traiter au point suivant.

4.4.2. "Prolonger un plan".

Nous avons vu dans les exemples qu'il peut être nécessaire de modifier la portion de plan qui est représentée, pour permettre de situer un point d'intersection, soit sur le plan horizontal, soit sur le plan de face.

En écrivant la procédure PROLONGER nous avons voulu résoudre ce problème une fois pour toutes. Nous lui passons 3 paramètres:

- Le numéro du plan dont il faut modifier la représentation.
- Le point vers lequel il faut "prolonger".
- Le plan dans lequel il faut "prolonger" (horizontal-vertical).

Le but de cette procédure est de modifier des sommets du parallélogramme représentant le plan, de telle manière que le point spécifié soit inclus dans cette représentation. Bien entendu, si ce point appartenait déja à la portion du plan représentée, nous ne la modifions pas (nous ne "réduisons" jamais).

Afin de faciliter la mise en oeuvre des calculs et de réduire le nombre de tests nous avons essayé d'avoir une représentation plus ou moins cohérente pour les 16 types de plan, ainsi que certaines propriétés communes. Par exemple, si le plan à représenter a une droite d'intersection avec le plan horizontal, le segment correspondant sera celui qui lie le sommet 1 au sommet 2. Cela permet de ne pas devoir distinguer trop de cas pour prolonger des plans de types différents.
Nous avons également défini une fonction INTERVALLE qui permet simplement de voir si un paramètre est compris entre deux autres paramètres. Cela permet d'allèger l'écriture des tests.
Venons en maintenant à la procédure en elle-même, nous l'avons divisée en 2 parties : selon la valeur du 3e paramètre (horizontal ou vertical) nous appelons PROLHOR et PROLVERT.
Dans PROLHOR nous distinguons encore 2 cas selon que A est nul ou non.
Si A est nul (voir ANUL) nous avons un plan d'équation BY + CZ = D. Ce plan est donc parallèle à l'axe X1 et la prolongation dans ce sens consiste simplement à modifier la première coordonnée de 2 sommets (1 et 4 ou 2 et 3 selon qu'il faut prolonger vers la gauche ou la droite). Si en plus B est nul, le plan est aussi parallèle à l'axe X2 et un prolongement dans le plan horizontal selon cet axe est également possible. De même, il suffit alors de modifier la 2e coordonnée de 2 sommets.
Si A est non nul, nous distinguerons les cas où B est ou non nul. S B $\neq$ 0 l'équation est AX + BY + CZ = D et si l'on modifie la 1ère coordonnée d'un sommet, il faut calculer la nouvelle valeur pour la 2e coordonnée.
De plus, la valeur de la 1ère coordonnée du second sommet modifié doit aussi être adaptée. Cela est réalisé en reportant le même décalage que sur la figure initiale, à partir du premier sommet modifié. Il ne reste alors plus qu'à calculer à partir de là, la 2e coordonnée du second sommet.
Par contre si B = 0, l'équation est AX + CZ = D et ce plan est parallèle à l'axe X2. Nous ne savons donc prolonger que dans ce sens et il suffit de modifier la 2e coordonnée de 2 sommets.
Dans PROLVERT, nous distinguons 3 cas: Si C= 0, si A et C $\neq$ 0 et si A = 0 et B et C $\neq$ 0.
Lorsque C est nul, le plan est parallèle à l'axe X3 et comme nous prolongeons dans le sens vertical, il n'y a qu'à modifier la 3e coordonnée de 2 sommets.
Sinon, lorsque A $\neq$ 0, la prolongation se fait sur base de l'équation AX + BY + CZ = D. Nous assignons alors à la 3e coordonnée du sommet à modifier la valeur de la 3e coordonnée du point et nous modifions sa 1ère coordonnée en conséquence. Nous faisons ensuite de même pour un second sommet.
Sinon, B $\neq$ 0 et l'équation est BY + CZ = D. Le principe est le même, mais le calcul différent puisque l'équation change.

4.4.3. Découpe générale de l'algorithme.

Nous avons vu que les méthodes à appliquer différaient selon les types des 2 plans en présence. Nous regardons d'abord si l'un ou l'autre des plans n'est pas impossible ou indéterminé (nous avons enregistré cela lors de la détermination du type du plan). Ensuite nous testons les paramètres pour voir si A, B et/ou C ne sont pas simultanément nuls dans les 2 équations.
Comme la structure de cet algorithme est claire, nous la reproduisons sous cette forme.

```
Si 1er plan impossible:
|ALORS envoyer message (pas de représentation)
|SINON si 2e plan impossible
        |ALORS envoyer message (pas de représentation)
        |SINON si 1er plan indéterminé
                |ALORS si 2e plan indéterminé
                        |ALORS message ("dblmt indéterminé")
                                représentation de tout l'espace
                        |SINON message ("plan 1 indéterminé")
                                représentation du 2e plan
                |SINON si 2e plan indéterminé
                        |ALORS message ("plan 2 indéterminé")
                                représentation du 1er plan
                        |SINON si A 1 = 0 et A 2 = 0
                                |ALORS si B1 = 0 et B2 = 0
                                        |ALORS [NIX1NIX2]
                                        |SINON si C 1 = 0 et C 2 = 0
                                                |ALORS [NIX1NIX3]
                                                |SINON [PAS DEX1]
                                |SINON si B 1 = 0 et B 2 = 0
                                        |ALORS si C 1 = 0 et C 2 = 0
                                                |ALORS [NIX 2NIX3]
                                                |SINON [PAS DEX2]
                                        |SINON si C 1 = 0 et C 2 = 0
                                                |ALORS [PAS DEX3]
                                                |SINON [CASGENERAL]
```

- A 1 = coëfficient A de la 1ère équation.
- B 2 = coëfficient B de la 2e équation.
- Les noms dans les encadrés sont ceux des procédures qui analysent ces différents cas.

4.4.4. Cas où 2 inconnues sont absentes.

voir NIX1NIX2 , NIX1NIX3 , NIX2NIX3.
Comme les 2 équations ne contiennent qu'une seule et même variable, les plans qu'elles représentent sont très particuliers.
Si le terme indépendant est nul, il s'agit du plan horizontal, de face ou debout, selon que la variable est Z, Y ou X.
S'il est non nul, le plan sera parallèle au précédent.
Ainsi pour chacune de ces procédures il n'y a que deux possibilités: ou bien les 2 plans sont confondus, ou bien ils sont parallèles.

4.4.5. Cas où 1 inconnue est absente.

voir PAS DEX1 , PAS DEX2 , PAS DEX3.
Nous savons que hormis la variable dont le nom apparaît dans celui de la procédure, chacune des 2 autres variables apparaît au moins dans une des deux équations.
Nous définissons toujours notre "première" équation (aux coëfficients A1,B1,C1,D1) comme l'équation où le coëfficient de la variable de plus petit ordre (X1 pour PAS DEX2 et X3, X2 pour PAS DEX1) est non nul, pour être sûr de ne pas diviser par 0 si nous devons calculer la position d'un point d'intersection.
Nous savons déjà que les 2 plans sont parallèles à l'axe correspondant à la variable absente.
Ainsi, si l'inclinaison des 2 plans est identique dans l'autre sens, les plans seront parallèles ou confondus.
Sinon, nous cherchons le point d'intersection entre les 2 plans , situé sur le plan de face, debout ou horizontal correspondant à la variable manquante. Nous "prolongeons" les 2 plans vers ce point. Nous déterminons un second point, qui ne se différencie du premier que par une seule coordonnée (celle qui correspond à cette variable absente), puisque la droite d'intersection est parallèle à cet axe.

4.4.6. Cas général.

Nous savons que chaque variable est présente dans au moins une des 2 équations.
Nous nous arrangeons donc pour que notre "première" équation contienne toujours un coëfficient de X1 non nul. ("A1" est toujours différent de 0).
En cours de raisonnement, nous serons souvent amenés à vérifier si 2 droites des 2 plans ne sont pas de même pente. Nous regarderons si les rapports entre les coëfficients correspondants à ces droites ne sont pas identiques. (Par exemple: pour vérifier si les 2 droites sur le plan horizontal, c.a.d. entre les axes X1 et X2, ne sont pas de même pente, nous testons si A2B1 - A1B2 = 0 ou encore, sauf division par 0, si A1/A2 = B1/B2). C'est à cet effet que nous calcu-

lons AB, AC, AD.

Pour des raisons de clarté et de concision nous allons présenter la suite sous forme forme algorithmique. (voir procédure CASGENERAL).

Si AB = 0 (→même pente pour les droites dans le plan horizontal.)

ALORS Si AC = 0 (→ idem dans le plan de face.)

ALORS si AD = 0

ALORS les 2 plans sont confondus.

SINON les 2 plans sont parallèles.

SINON si AD = 0

ALORS la droite d'intersection est la droite du premier plan, située sur le plan horizontal.

SINON nous cherchons un 1er point d'intersection dans le plan de face, nous "prolongeons" les 2 plans. Nous calculons une seconde extrémité de la droite d'intersection, (celle-ci est parallèle à la droite du 1er plan sur le plan horizontal) de telle manière qu'elle "couvre" tout le schéma. (voir point 2.2.2 page 22 pour exemple).

(voir ABNUL)

SINON Si AC = 0

ALORS si AD = 0

ALORS la droite d'intersection est la droite du 1er plan située sur le plan de face.

SINON nous cherchons le 1er point de la droite d'intersection dans le plan horizontal. Après "prolongations", nous déterminons une seconde extrémité de la droite d'intersection (parallèle à la droite du 1er plan dans le plan de face) et nous "prolongeons".

(voir ACNUL)

SINON nous cherchons le point d'intersection sur le plan horizontal et nous prolongeons les plans.

si AD = 0

ALORS comme le 1er point est situé sur l'axe X1, il appartient à la fois au plan horizontal et au plan de face. Nous calculons notre seconde extrémité comme étant le point de la droite d'intersection situé à la même hauteur que la droite d'un des plans la plus élevée.

(voir ADNUL)

SINON nous cherchons le 2e point comme l'intersection de 2 droites dans le plan de face et nous prolongeons.

(voir ADNONNUL)

nous regardons ensuite si les 2 points trouvés sont bien sur une "droite limite" d'un plan. (voir exemple page 24)
Nous utilisons la fonction LIMITE qui regarde si un point donné est situé sur une des 4 droites représentant le plan.
Si un des points est situé "au milieu du parallèlogramme" nous modifions sa position pour qu'il se situe sur une droite limite.
(voir CHINT1 , CHINT2)

(voir ACNONNUL)

4.5. PRESENTATION DU SYSTEME COMPLET.

(voir SYST 33)

Les méthodes expliquées pour la superposition de 2 plans seront encore utilisées ici. C'est la procédure INTERPLAN qui déterminera les portions de plan représentées pour la superposition des 3 plans.
Nous appliquerons trois fois cette procédure: d'abord sur les plans d'équations 1 et 2, ensuite sur les plans 1 et 3 et enfin sur les plans 2 et 3.
Comme nous appliquons chaque fois cette procédure sur les sommets tels qu'ils ont été laissés par l'appel précédent, les portions de plan pourront être beaucoup plus grandes.
Cela est nécessaire pour que chacun des 3 plans soit représenté plus ou moins dans le même ordre de grandeur.

Ainsi pour présenter un système 3 x 3 à l'utilisateur nous présentons d'abord le 1er plan, le 2e plan et leur superposition (voir DEUX PLANS).
Ensuite nous faisons de même pour les couples de plans (1,3) et (2,3).
Enfin nous déterminons les portions de plans à représenter en appliquant succesivement la procédure INTERPLAN.
Nous mémorisons les différentes extrémités des segments d'intersection dans le tableau PTINTER. A ce propos il subsiste un petit problème que nous n'avons pas eu le temps de résoudre : il se peut que certains de ces points d'intersection ne soient pas situés sur une droite limite d'un des plans (car leur portion représentée peut avoir été modifiée).

5. AFFICHAGES A L'ECRAN.

En "construisant" ce système nous nous sommes rendus compte que la communication avec l'utilisateur, via l'écran du micro-ordinateur pouvait être vue de 2 façons:

1. afficher des textes à l'écran: textes d'explication, message d'erreur, message de rappel de choix ...
2. afficher des menus, saisir la réponse de l'utilisateur et la valider.

Après avoir défini le contenu de ces textes et menus nous nous sommes aperçus que :

- l'on pouvait distinguer 2 types de textes : nous avons d'une part des textes d'explication assez longs (l'écran entier) et d'autre part des messages beaucoup plus courts (quelques lignes).
- Chaque menu, même si le nombre de choix était variable (et limité à 5), avait la même structure.
  - Un titre explicatif.
  - n propositions de choix (n est compris entre 2 et 5), avec le numéro correspondant.
  - Une demande de réponse du type : "quel est votre choix (1 - n) ?".

Nous avons donc défini une structure de données de nom TEXTE qui peut contenir jusqu'à 24 lignes de 40 caractères (surface maximale de l'écran) et une autre de nom MESSAGE qui peut en contenir 9. Pour chaque texte ou message souhaité, nous avons une procédure d'initialisation au contenu souhaité (voir INIT 1 à 5 pour les textes et INIC 1 à2 et INIER 1 à 5 pour les messages).
Pour afficher un texte, nous appelons AFFTEXTE avec comme paramètre un numéro prédéfini correspondant à l'initialisation souhaitée.
Pour les messages, la procédure s'appelle AFFMESSAGE.
Ces 2 procédures affichent succesivement les différentes lignes.
En ce qui concerne les menus, chaque titre et explication de choix pouvait comporter 2 lignes. Nous définissons donc TITRE comme un tableau de 2 chaînes de caractères et MENU comme un tableau de 5 x 2 chaînes de caractères, puisque nous n'offrons que 5 choix au maximum par menu.
Nous crèons une routine d'initialiastion pour chaque titre, et un autre pour les différents choix de chaque menu. (voir INITIT 1 à 8 et INIMENU 1 à 8).
Nous appelons la routine AFFMENU avec comme paramètres le numéro du menu souhaité et le nombre de choix possibles.
Elle affiche le titre , puis chaque choix (précédé de son numéro) et enfin demande une réponse. Celle-ci est testée par la procédure VALIDATION qui vérifie si la réponse est bien un chiffre entre 1 et le nombre de choix et qui demande un autre

choix tant que ce n'est pas le cas.

Cette procédure fournit comme résultat la réponse (validée) de l'utilisateur.

Remarque : En ce qui concerne les menus 2 et 5, nous ne les affichons pas en haut de l'écran, mais bien après un message qui rappelle le choix effectué. C'est la raison du test rencontré après les initialisations dans AFFMENU.

6. INTRODUCTION DES SYSTEMES ET DES REPONSES.

Nous avons regroupé l'introduction des systèmes et celle des réponses en un seul module, parce que nous utilisons la même démarche d'analyse dans les deux cas .

6.1. INTRODUCTION ET SAISIE DES SYSTEMES D'EQUATIONS LINEAIRES.

Nous voulons imposer peu de contraintes à l'utilisateur.
Nous ne lui demandons pas d'introduire les coëfficients de ses équations un par un en réponse à une invitation du genre "Veuillez introduire le coëfficient de la 3e inconnue de la 2e équation".
Ce que nous lui proposons, c'est de rentrer son sytème en entier, comme il a l'habitude de le traiter.
Le résultat sera que les coëfficients du sytème d'équations linéaires de dimension m * n sont enregistrés dans les m premières lignes et n + 1 premières colonnes d'une matrice (la n + 1 ^{ème} contient les termes indépendants).
Notons enfin que dans la pratique les exercices demandés aux élèves sont le plus souvent limités à l'ordre 3 x 3. De temps à autre ils ont un système de 4 x 4 à résoudre, mais pratiquement jamais de 5 x 5. Ainsi nous limiterons les systèmes admis à 5 équations et 5 inconnues.

6.1.1. Conventions de forme pour les équations.

Nous avons essayé de choisir des conventions simples, habituelles et imposant assez peu de contraintes.
Nous les avons définies au point 2.4.1.(pages 29 à 32).

6.1.2. Matrice de transition et matrice d'actions.

Une matrice de transition permet, à partir d'un état courant et d'un caractère lu, de passer à un état suivant. Chaque état est caractérisé par le type de caractère qu'il admet comme continuateur et par les états suivants dans lesquels il peut passer.
Une matrice d'action associe à un couple état-caractére lu, une action. Chaque action a pour but de permettre un enregistrement correct de la signification du caractère lu.
La méthode que nous utilisons est la suivante: au départ nous nous trouvons dans l'état initial. En fonction du premier caractère que nous lisons, nous effectuons une action (nous savons laquelle par la matrice des actions) et nous passons à

un état suivant (défini par la matrice de transition). Cet état suivant devient état courant. En fonction de celui-ci et du caractère suivant nous effectuons de nouveau une action et nous passons à un état suivant, etc... etc...

Cette méthode est très intéressante pour 2 raisons :

1. Elle oblige à distinguer tous les cas possibles car pour construire la matrice de transition, il faut nécessairement les examiner tous. Cela diminue donc les risques d'oubli.
2. La détermination des actions à effectuer est facilitée par le fait que dans chacun des cas on sait exactement où on en est arrivé.

6.1.3. Construction de la matrice de transition.

D'après nos conventions nous pouvons définir plusieurs types de caractères, Nous avons défini le tableau suivant :

Caractère lu :	Type :
Chiffre de 0 à 9	1
Barre de fraction "/"	2
Point "." ou virgule ","	3
Signe "+"	4
Signe "-"	5
Lettre "X"	6
Signe "="	7
RETURN	8
Blanc	10
Flèche "←"	11
N'importe quel autre caractère	9

Nous avons donc défini 11 types (A chacun d'eux nous associons un numéro). La matrice comprendra ainsi 11 colonnes, le nombre de lignes dépendra des successions d'états possibles.

Appelons notre état initial (début de chaque ligne) l'état 0.

Il faut remplir la première ligne de la matrice :

(Type)	1	2	3	4	5	6	7	8	9	10	11
(Etat)0											

Dans chacune des cases, il faut placer l'état dans lequel on passera si on lit un caractère du type correspondant.

Si le caractère lu est de type :

1. (chiffre de 0 à9) : Cela signifie que le signe a été omis (donc assimilé à +) et que le coëfficient commence par ce chiffre. Nous passons à l'état 1.
2. (/) : Ce caractère n'a aucun sens dans le contexte. Nous demandons à l'utilisateur d'en taper un autre. Nous ne changeons pas d'état, nous restons dans la situation précédente, comme si l'utilisateur n'avait encore rien tapé.
3. (. ou ,) : L'utilisateur n'a ni rentré de signe, ni de partie entière. Nous passons à l'état 5.
4. (+) : Nous passons à l'état 2. (Celui-ci ne diffère que très peu de l'état 0, la seule différence est que nous n'admettons pas un 2e signe).
5. (-) : Nous passons aussi à l'état 2. Le fait d'avoir lu un signe (qu'il soit + ou -) nous amène dans la même situation.
6. (X) : Il n'y a ni signe, ni coëfficient (donc "+1"). Nous passons à l'état 3.
7. (=) : Aucun sens ici (voir point 2.)
8. (RETURN) : L'utilisateur signale qu'il a terminé d'introduire son système. Nous passons donc à l'état final (16).
9. (caractère invalide) : Nous ne changeons pas d'état.
10. (blanc) : Ce caractère est permis n'importe où, mais il ne modifie pas la la situation courante. Nous restons donc dans le même état.
11. (←) : Comme il est permis d'effacer la ligne en cours à n'importe quel moment, nous revenons en début de ligne (c. a. d. à l'état 0).

Nous obtenons donc la 1ère ligne de notre matrice d'états. (TT voir INIT2).

(Type)	1	2	3	4	5	6	7	8	9	10	11
(Etat 0)	1		5	2	2	3		16		0	0

Nous avons suivi une raisonnement semblable pour chaque nouvel état et nous avons obtenu la matrice suivante.

ETAT \ CAR. LU	0..9	/	. ,	+	-	x	=	RET	INV	Blanc	←
	1	2	3	4	5	6	7	8	9	10	11
0	1		5	2	2	3		16		0	0
1	1	4	5			3				1	0
2	1		5			3				2	0
3	6									3	0
4	7									4	0
5	8									5	0
6				2	2		9			6	0
7	7					3				7	0
8	8					3				8	0
9	10		13	11	11					9	0
10	10	12	13					16		10	0
11	10		13							11	0
12	14									12	0
13	15									13	0
14	14							16		14	0
15	15							16		15	0

Nous pouvons relever quelques particularités dans cette matrice :

- La 10e colonne (correspond au blanc) : elle reprend en fait la numérotation des états. C'est normal puisque le blanc, qui est admis à tout endroit, ne modifie en rien la situation précédente.
- La 11e colonne (correspond à "←"): elle ne comprend que des 0. De nouveau ce caractère est admis partout. Comme il a pour effet d'effacer la ligne, il faut chaque fois se repositionner à l'état initial (ou 0).

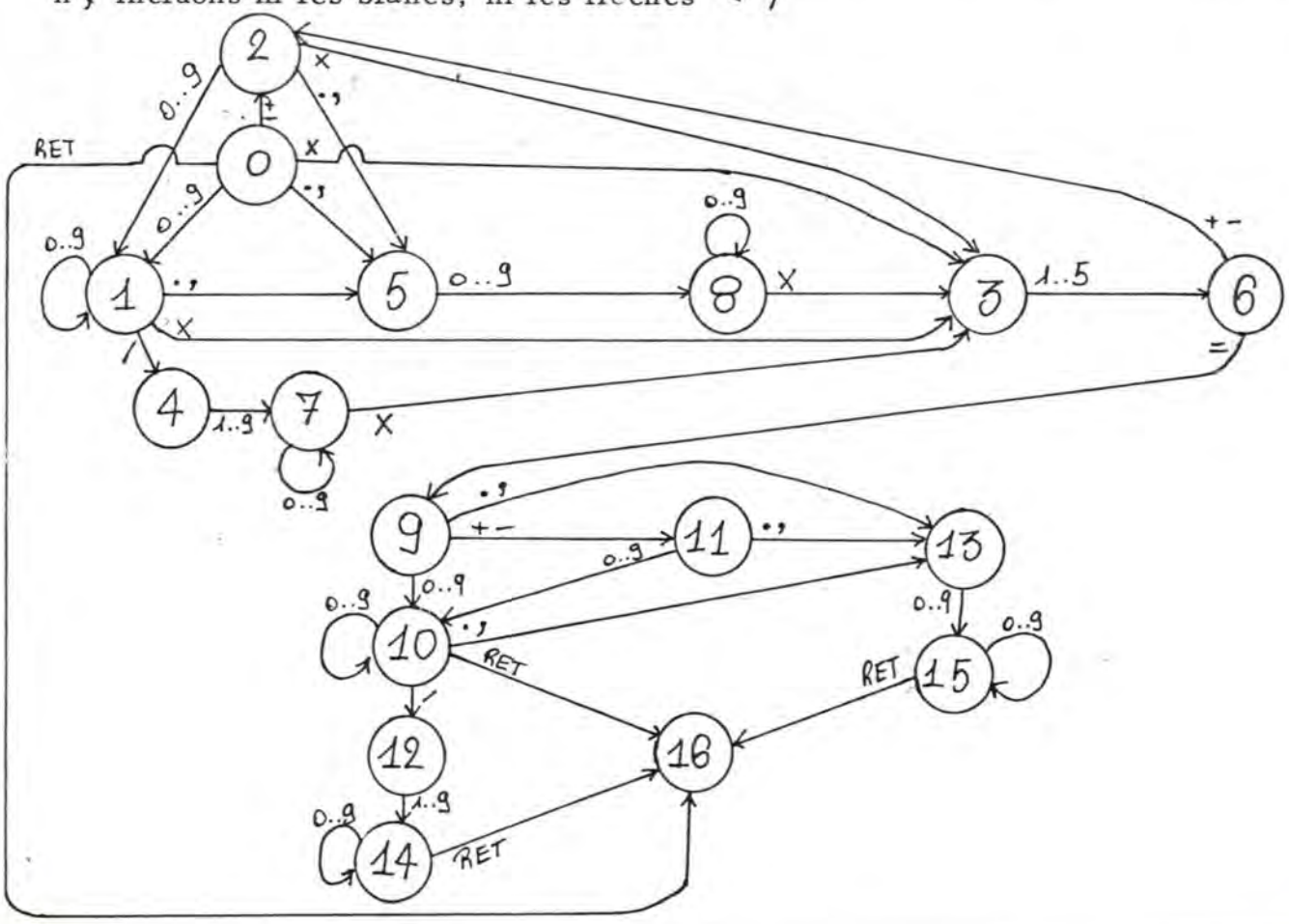
- Les colonnes 4 et 5 sont semblables. Nous avons vu que le fait de lire un "+" ou un "-" conduisait à un même état.
  L'intérêt de garder ces 2 colonnes se révèlera dans la matrice des actions. Celles-ci seront en effet différentes selon "+" ou "-".

- Cette deuxième remarque peut également s'appliquer aux couples de lignes (4,5) , (7,8) , (12,13) et (14,15).

- Nous constatons aussi que chaque fois qu'un RET est valide (voir colonne 8) nous passons invariablement à l'état 16, que la ligne soit finie ou que le système soit terminé. Cela est dû au fait que nous procédons à l'analyse ligne par ligne et que nous analysons une ligne tant qu'un indicateur de fin d'analyse n'est pas vrai. Celui-ci est évidemment positionné si un RET est lu en début de ligne ou si 5 équations ont déja été entrées.

- Nous n'avons défini qu'un seul type pour les chiffres. Or dans certains cas, même si on attend un chiffre, il ne sont pas tous admis : un indice ne peut appartenir qu'à l'intervalle (1..5), le premier chiffre d'un dénominateur ne peut être 0.
  Afin de ne pas devoir définir plus de types (et donc d'aggrandir les matrices), nous avons effectué ces contrôles au niveau des actions correspondantes.

Ci-dessous nous schématisons les différentes transitions possibles. (N.B. Nous n'y incluons ni les blancs, ni les flèches '←')



Nous caractérisons maintenant les différents états par la situation qu'ils représentent :

- Etat 0 : C'est l'état initial, où nous attendons le premier caractère d'une équation.
- Etat 1 : Nous avons commencé à lire un coëfficient.
- Etat 2 : Il s'agit presque d'un état initial, nous avons seulement lu un signe.
- Etat 3 : Nous venons de lire un "X" Donc nous attendons uniquement un indice.
- Etat 4 : Nous venons de lire une barre de fraction ; nous devons donc trouver le début d'un dénominateur, c'est à dire un chiffre.($\neq 0$)
- Etat 5 : Nous venons de lire un point ou une virgule : nous attendons le début d'une partie décimale, donc un chiffre.
- Etat 6 : Nous venons de lire l'indice d'une inconnue. Il ne peut donc suivre qu'un signe "+", "-" ou "=".
- Etat 7 : Nous avons commencé à lire le dénominateur d'un coëfficient fractionnaire. Soit ce dénominateur continue (par un chiffre), soit il est terminé et on rencontre un "X"
- Etat 8 : Nous sommes en train de lire la partie décimale d'un coëfficient. Soit cette partie décimale continue (par un chiffre), soit elle est terminée et on rencontre un "X"
- Etat 9 : Nous venons de lire un signe "=" Nous attendons donc le début d'un terme indépendant (un chiffre, un point ou une virgule, un signe).
- Etat 10 : Nous sommes en train de lire le terme indépendant. Soit il se continue (chiffre, "/" ou "." (",")). Soit il se termine (par un RETURN). Dans ce cas l'analyse de cette ligne est terminée.
- Etat 11 : Nous venons de lire un signe ("+" ou "-") comme premier élément d'un terme indépendant. Celui-ci peut se continuer par un chiffre ou un "." (",").
- Etat 12 : Nous venons de lire une barre de fraction : Nous attendons le dénominateur du terme indépendant, donc uniquement un chiffre.($\neq 0$)
- Etat 13 : Nous venons de lire un point ou une virgule, Nous allons avoir la partie décimale d'un terme indépendant, donc uniquement un chiffre.

- Etat 14 : Nous sommes en train de lire le dénominateur du terme indépendant. Soit celui-ci se continue (par un chiffre), soit il se termine (RETURN).

- Etat 15 : Nous sommes en train de lire la partie décimale du terme indépendant. Si elle continue nous avons un chiffre. Sinon, elle est terminée et nous avons un RETURN.

-Etat 16 : C'est l'état final. Lorsqu'on le rencontre, c'est que l'analyse de la ligne est terminée.

Nous savons maintenant comment passer d'état en état. Comme ce passage nécessite la validité des caractères lus,nous sommes sûrs de n'accepter que des équations qui ont un sens. Il reste maintenant à les enregistrer.

6.1.4. Construction de la table des actions.

Nous allon associer à chaque couple (état, caractère) une action. L'élément (m,n) de cette matrice est la référence à une action à effectuer lorsque dans l'état m, nous lisons un caractère de type n.
Toutes les cases de la matrice seront cette fois occupées :

- Si le caractère n'a pas de sens : nous envoyons un message d'erreur et nous demandons qu'il introduise un autre.
- Sinon nous devons mémoriser le "sens" de ce caractère.

Nous définissons ces différentes actions plus loin (voir point 6.1.6), car avant cela nous définissons les structures de données qui vont permettre l'enregistrement des coëfficients et la structure de la procédure qui va mettre ces deux matrices en oeuvre.

6.1.5. Comment enregistrer les coëfficients ?

Les coëfficients peuvent être entiers, fractionnaires ou décimaux. Comme nous l'avons déjà dit nous essaierons toujours d'enregister chaque coëfficient à l'aide de deux entiers, un numérateur et un dénominateur :

Exemples : 3 sera enregistré comme 3/1.
2/3 sera enregistré comme 2/3.
0.11 sera enregisté comme 11/100.

A cet effet nous définissons un record FRACTION, composé d'un numérateur et d'un dénominateur. (Nous définissons plus loin des opérations sur ces fractions). Nous voulons enregister ces coëfficients dans une matrice ; pour cela nous définissons un type MATRICE qui est un tableau à 2 dimensions de FRACTIONS. Cette matrice comprend 5 lignes et 6 colonnes. Mais selon la taille effective

du système (m x n), nous enregistrons les coëfficients dans la partie supérieure gauche de la matrice(les m premières lignes et n+1 premières colonnes).
Il est encore nécessaire de définir un vecteur qui contiendra temporairement, les termes indépendants, parce que c'est à la fin de l'introduction que nous savons dans quelle colonne de la matrice il faut enregistrer les termes indépendants. Nous ne connaissons pas cette place plus tôt, car l'utilisateur n'est pas obligé d'introduire toutes les inconnues dans chaque équation.
Que se passe-t-il si le numérateur ou le dénominateur est trop grand pour être enregistré comme entier (capacité maximale = + ou - 32.000)? Nous allons traiter ce coëfficient comme un réel. Nous définissons aussi une matrice et un vecteur (termes indépendants) pour les réels.
Nous parlerons de mode "réel" ou "flottant" si tous les coëfficients sont enregistrés en réels. Sinon, nous parlerons de mode "fraction". Comme notre souci sera toujours d'essayer d'éviter le passage au mode réel, nous serons très prudent avant d'y passer définitivement.
Supposons en effet que l'utilisateur introduise par erreur un coëfficient ou un terme indépendant qui dépasse la capacité des entiers. S'il efface ce nombre (soit par la flèche "←", soit en corrigeant la ligne à la fin de l'introduction), il se peut qu'il n'y ait plus de raison de travailler en mode réel. Afin d'éviter cela nous considérons toujours que la ligne qui va être introduite le sera en mode fraction, même si une ligne précédente a déjà entrainé un passage en mode réel. Si dans cette ligne, un nombre entraîne le passage en réel, toute cette ligne est enregistrée en mode réel, mais uniquement celle-là. Ce qui revient à dire que si l'utilisateur efface cette ligne ou la corrige par après, lorsqu'il va la réintroduire, nous attendrons une ligne en mode fraction.
A chaque ligne correspondra un indicateur qui permettra de voir dans quel mode la ligne a été enregistrée. C'est le rôle du vecteur OK. (OK [i] est vrai implique que la ligne i a été enregistrée en fraction).
Donc pour déterminer si, après la correction, il y a lieu ou non de passer en mode réel, nous regardons si au moins un indicateur OK [i] est faux.
Nous pouvons donc expliquer la structure de la procédure principale (voir procédure INTRODUCTION). Il faut bien sûr réaliser les initialisations des matrices.
Ensuite nous analysons les équations une par une (voir description d'ANALIGNE au point 6.1.6). Nous introduisons les termes indépendants dans la matrice ; il s'agit simplement de recopier ceux-ci, stockés dans un vecteur, à la bonne place dans la matrice (voir INTTI).
Il faut ensuite permettre à l'utilisateur de corriger son sytème jusqu'à ce qu'il soit satisfait. (voir CORRIGER et point 6.1.7)
Enfin il ne reste plus qu'à voir si on peut travailler en mode fraction ou si le passage en mode réel est obligatoire (voir DETERMINERMODE).

6.1.6. La procédure ANALIGNE (+ description des actions).

Elle met les deux matrices en oeuvre. Pour chaque ligne nous partons de l'état 0 (avec signe initial à "+") et tant que nous ne sommes pas en fin de ligne (état 6) nous bouclons :

- -aller chercher le caractère suivant (voir NEXTCAR) et renvoyer son type.
- -effectuer l'action correspondant au numéro trouvé dans la matrice des actions. (TA voir INIT 5).
- -si l'action ne consistait pas à "refuser" le caractère, l'écrire, avancer d'une position sur la ligne et passer à l'état suivant.

Nous spécifions maintenant la table des actions et chacune d'elles. Chaque case contient le numéro d'action : nous appelons les actions dans le "CASE" de la procédure.

CAR. LU / ETAT	0..9	/	. ,	+	-	x	=	RET	INVA	Blanc	←
	1	2	3	4	5	6	7	8	9	10	11
0	1	99	20	2	3	4	99	18	99	0	98
1	5	6	7	99	99	8	99	99	99	0	98
2	1	99	20	99	99	4	99	99	99	0	98
3	9	99	99	99	99	99	99	99	99	0	98
4'	21	99	99	99	99	99	99	99	99	0	98
5	10	99	99	99	99	99	99	99	99	0	98
6	99	99	99	2	3	99	19	99	99	0	98
7	14	99	99	99	99	11	99	99	99	0	98
8	12	99	99	99	99	13	99	99	99	0	98
9	1	99	20	2	3	99	99	99	99	0	98
10	5	6	7	99	99	99	99	15	99	0	98
11	1	99	20	99	99	99	99	99	99	0	98
12	21	99	99	99	99	99	99	99	99	0	98
13	10	99	99	99	99	99	99	99	99	0	98
14	14	99	99	99	99	99	99	16	99	0	98
15	12	99	99	99	99	99	99	17	99	0	98

Nous allons spécifier le but de ces différentes actions, mais auparavant nous allons définir le rôle des variables intermédiaires utilisées.

-NOMBRE : il s'agit d'un entier qui sert de tampon général chaque fois qu'une partie du coëfficient est entière.

- NBREEL : joue le même rôle que NOMBRE, mais en réel.

- NUM et DEN : sont deux entiers qui servent de tampon pour le dénominateur et le numérateur d'une fraction.

- NUMREEL : sera utilisé comme tampon pour le numérateur, s'il y a eu un passage en réel. (DENREEL n'est pas nécessaire, car en mode réel on n'utilise plus de fraction!)

- PARTENT : sera utilisé comme tampon pour la partie entière. Il s'agit d'un entier. (Il n'y a pas d'équivalent en réel, car alors les décimales sont enregistrées au fur et à mesure).

- CPTDEC : il s'agit du compteur de décimales. Il sert à pouvoir transformer un coëfficient décimal en fraction.

- SIGNE : il peut prendre la valeur "+1" ou "-1". Nous multiplions chaque nombre par cette variable avant de l'enregistrer pour tenir compte du signe.

- ERREUR :vrai si le caractère est erroné.

ACTION N° 0 .

Correspond à ne rien faire (exemple : après un blanc, il n'y a pas d'action spécifique.)

ACTION N° 1.

Cette action est exécutée pour initialiser un nombre, car il s'agit du premier chiffre que l'on lit. Il suffit de transformer le caractère lu en numérique et selon le mode de la ligne, l'enregistrer dans NOMBRE ou NBREEL.

ACTION N°2.

Nous venons de lire un "+". Donc le coëfficient ou le terme indépendant sera positif. Nous positionnons SIGNE à "+1".

ACTION N°3.

Nous venons de lire un "-". Nous positionnons SIGNE à "-1".

ACTION N°4.

Nous venons de lire "X" soit en début de ligne, soit après un signe "+" ou "-". Donc le coëfficient a été omis : sa valeur absolue est 1. Donc nous mémorisons la valeur de SIGNE (+1 ou -1).

ACTION N°5.

Nous venons de lire un chiffre qui continue un nombre. Nous multiplions par 10

la partie déja enregistrée et nous ajoutons le chiffre. Nous vérifions si cela ne provoque pas un dépassement de capacité(voir CAPACITE). Si c'est le cas il faut transformer cette ligne de fractions en réels (voir LIGRELLE).

ACTION N°6.

Nous venons de lire une barre de fraction. Cela signifie que le numérateur est terminé. Selon le mode nous enregistrons la valeur de NOMBRE ou NBREEL, miltiplié par SIGNE, dans NUM ou NUMREEL.

ACTION N°7.

Nous venons de lire un . ou , . La partie entière est donc terminée. En mode entier nous l'enregistrons dans PARTENT. En mode réel nous le laissons dans NBREEL, car nous le modifieronsau fur et à mesure de l'apparition des décimales.

ACTION N°8.

Nous venons de lire un "X" après un nombre, qui constitue donc un coëfficient entier. Selon le mode, nous l'enregistrons dans NUM (et DEN = 1) ou dan NBREEL.

ACTION N°9.

Nous venons de lire un indice (car le caractère précédent = "X"). Celui-ci doit être compris entre 1 et 5. S'il ne l'est pas nous envoyons un message d'erreur et nous positionnons l'indicateur d'erreur. Sinon, nous connaissons l'endroit de la matrice où il faut enregistrer le coëfficient et nous le faisons.
De plus nous vérifions s'il ne s'agit pas du plus grand indice déjà rencontré.

ACTION N° 10.

Nous venons de lire le premier chiffre d'une partie décimale. Nous initialisons CPTDEC à 1. Si nous sommes en mode fraction (la partie entière est mémorisée dans PARTENT), nous initialisons NOMBRE à ce chiffre. Sinon, nous enregistrons directement la décimale dans NBREEL.

ACTION N° 11.

Nous venons de lire un "X" après un coëfficient fractionnaire. En mode fraction, NUM contient le numérateur et NOMBRE le dénominateur. Il suffit d'affecter la valeur de NOMBRE à DEN. En mode flottant, le numérateur avait été stocké dans NUMREEL. Il suffit d'affecter à NBREEL le résultat de la division de NUMREEL par NBREEL (qui contient le dénominateur).

ACTION N°12.

Nous venons de lire un chiffre qui fait partie d'une partie décimale. Nous incrémentons CPTDEC de 1.
En mode fraction, on complète la partie décimale et on véréfie si elle ne dépasse pas la capacité maximale. Si oui, on transforme la ligne en réels et on donne sa valeur exacte à NBREEL. Sinon on enregistre la partie décimale dans NOMBRE.
En mode réel, on complète ce coëfficient décimal dans NBREEL.

ACTION N°13.

Nous venons de lire un "X" après un coëfficient décimal. En mode fraction, il faut transformer ce coëfficient en fraction grâce à PARTENT, NOMBRE et CPTDEC et à la fonction EXPO qui permet de calculer un exposant. Si à un moment, il y a dépassement de capacité il faut bien sûr changer de mode et donner une valeur correcte à NBREEL.

En mode réel, il n'y a rien à faire, car le coëfficient décimal a été enregistré directement comme tel.

ACTION N° 14.

Cette action est pratiquement semblable à l'action 5. La seule différence est que ce nombre est ici un dénominateur. S'il surgit un dépassement de capacité il est nécessaire de mémoriser le numérateur entier (NUM) en NUMREEL.

ACTION N°15.

Nous venons de lire un RETURN juste après un terme indépendant entier. Nous allons donc enregistrer celui-ci. Nous mémorisons aussi la valeur courante de l'indice maximum. (voir aussi action 98).

ACTION N° 16.

Même chose que l'action 15, si ce n'est que le terme indépendant est fractionnaire.

ACTION N° 17.

Nous venons de lire un RETURN après un terme indépendant décimal. Il faut donc l'enregistrer (même principe de calcul que dans l'action n° 13).

ACTION N° 18.

Nous venons de lire un RETURN en début de ligne. C'est une convention qui signale la fin du système. Nous positionnons donc l'indicateur de fin de système à vrai.

ACTION N° 19.

Nous venons de lire le signe "=". Il est possible que le terme indépendant n'ait pas de signe explicite. Donc il faut positionner SIGNE à "+1".

Remarque: Cette action est la même que l'action 2. Nous en avons fait une action séparée parce que logiquement la situation est différente.

ACTION N°20.

Nous venons de lire un . ou une , sans partie entière explicite. Il faut donc l'initialiser à 0.

ACTION N° 21.

Nous venons de lire le premier chiffre après une barre de fraction. Comme le dénominateur ne peut être nul, il faut que ce chiffre soit non nul.

ACTION N° 98.

Elle correspond à la touche d'effacement. Nous nous repositionnons en début de ligne, nous remettons cette ligne à zéro (voir LIGAZERO) et nous repo-

sitionnons le booléen OK à vrai pour essayer d'enregistrer la ligne en fraction. Nous comparons aussi la valeur courante de l'indice maximum rencontré (MAXIND) avec la valeur qu'il a dans cette ligne (INDMAX). Si INDMAX est supérieur à MAXIND, cela signifie que dans la ligne qu'on efface nous avons rencontré une inconnue d'indice supérieur à celles rencontrées dans les équations précédentes. On rend donc à INDMAX sa valeur courante précédente.

ACTION N° 99.

Il s'agit de l'action à effectuer lorsque le caractère ne doit pas être pris en considération parce qu'il est erroné.

Nous positionnons ERREUR à vrai. Ainsi nous n'affichons pas ce caractère, nous restons à la même place sur la ligne et nous ne changeons pas d'état.

De plus, nous envoyons un message à l'utilisateur pour lui montrer quels sont les types de caractère admis. Nous utilisons à cet effet une table de message (TM, voir INIT4) qui est directement déduite de la table des transitions. En effet pour chaque état nous connaissons grâce à celle-ci le type de caractères suivants admis.

6.1.7. Remarque sur la procédure CORRIGER.

Il y a un point important à signaler. Si en modifiant une ligne on introduit une inconnue d'indice supérieur aux inconnues entrées dans les équations précédentes nous :

1. Remettons tous les coëfficients de cette inconnue à 0 (sauf celui de la nouvelle ligne).
2. Remettons les termes indépendants à leur place (celle-ci a été modifiée, puisqu'il y a une inconnue supplémentaire).

Cette approche peut sembler lourde : en effet, si nous attendions la fin de la correction avant d'introduire les termes indépendants, ce problème n'existerait pas. Si nous avons malgré tout choisi cette méthode, c'est pour pouvoir utiliser la procédure d'impression de matrice pour la présenter.

Sinon, nous aurions dû définir, en plus, une autre routine d'impression, tenant compte aussi du vecteur des termes indépendants.

6.2. INTRODUCTION ET SAISIE DES REPONSES.

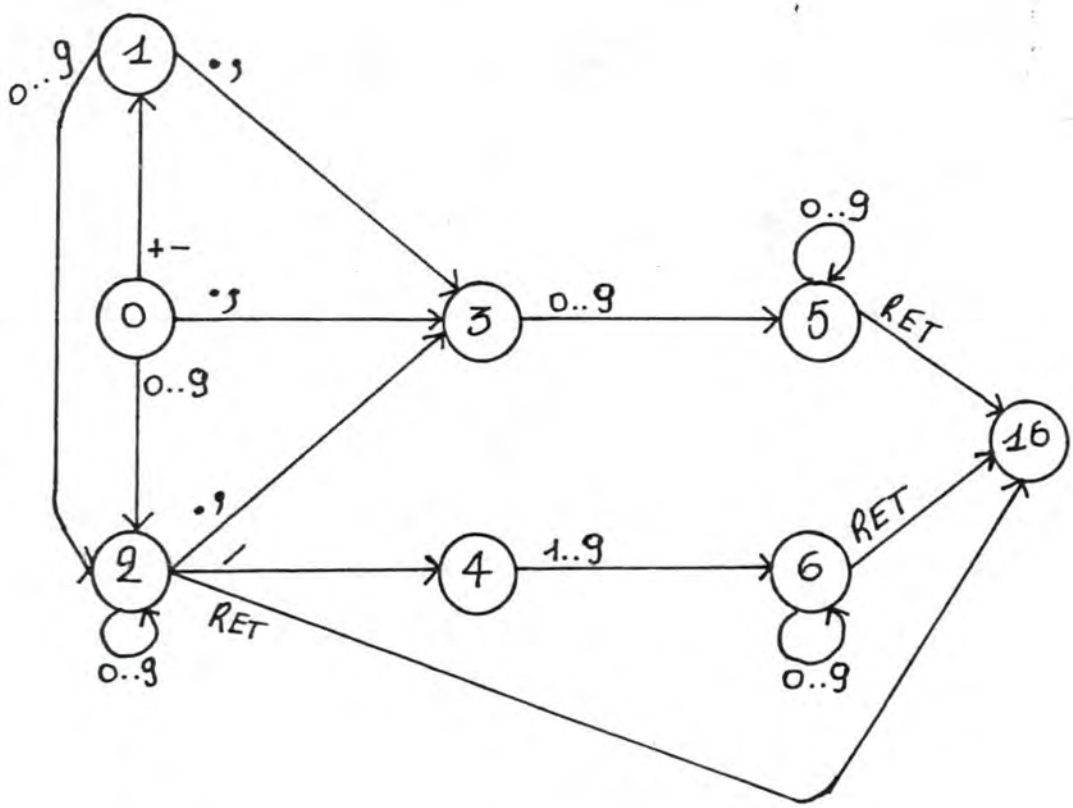
Nous n'enregistrons que des solutions uniques. Nous proposons, pour chaque variable, " X =", à l'utilisateur. Il ne lui reste alors plus qu'à compléter par la valeur qu'il a effectivement trouvée pour cette inconnue. Cette valeur peur avoir la forme d'un coëfficient ou d'un terme indépendant. Nous n'avons

donc besoin que d'une matrice de transition et d'une matrice d'action réduites. De plus, la plupart des actions sont identiques. Seules sont modifiées les actions qui consistent en l'enregistrement final des valeurs.

6.2.1. La matrice de transitions (voir TT2 dans INIT5).

	0..9	/	. ,	+	-	x	=	RET	INVA	Blan	←
	1	2	3	4	5	6	7	88	9	10	11
0	2		3	1	1					0	0
1	2		3							1	0
2	2	4	3					16		2	0
3	5									3	0
4	6									4	0
5	5							16		5	0
6	6							16		6	0

Nous pouvons schématiser comme suit :



6.2.2. La matrice d'actions. (voir TA2 dans INIT 6).

	0..9	/	. ,	+	-	x	=	RET	INVA	Blanc	←
Type	1	2	3	4	5	6	7	8	9	10	11
Etat											
0	1	99	20	2	3	99	99	99	99	0	98
1	1	99	20	99	99	99	99	99	99	0	98
2	5	6	7	99	99	99	99	31	99	0	98
3	10	99	99	99	99	99	99	99	99	0	98
4	21	99	99	99	99	99	99	99	99	0	98
5	12	99	99	99	99	99	99	32	99	0	98
6	14	99	99	99	99	99	99	33	99	0	98

Toutes ces actions ont été décrites pour l'introduction des systèmes, sauf 31, 32 et 33, qui enregistrent les valeurs mémorisées. Pour cette mémorisation, nous avons défini 2 vecteurs :

- SFUNIQUE qui est un vecteur "SOLENTIER."
- SRUNIQUE qui est un vecteur "SOLREEL."

(Nous trouvons la définition de SOLENTIER et SOLREEL en annexe page).

ACTION N° 31.

Nous venons de lire une valeur entière. Si elle peut être enregistrée comme une fraction, nous enregistrons (SIGNE * NOMBRE) dans SFUNIQUE [i]. NUM (et .DEN = 1). Sinon nous enregistrons (SIGNE * NBREEL) comme SRUNIQUE[i]. SOLUTION.

ACTION N° 32.

Nous venons de lire une valeur décimale. Le principe est le même que pour l'action 13 (voir page), sauf pour l'enregistrement.

ACTION N° 33.

Nous venons de lire une valeur fractionnaire. Nous enregistrons en fraction ou en réel, selon que OK [i] est vrai ou faux. (Même principe que l'action 11, page).

7. RESOLUTION DES SYSTEMES D'EQUATIONS LINEAIRES.

Nous présentons ce point en trois parties:

- Simplifier les coëfficients de la matrice de départ.
- Réaliser l'élimination des variables.
- Interprèter la matrice ainsi réduite et présenter les résultats.

7.1. SIMPLIFICATION DE LA MATRICE DE DEPART (voir SIMPLIMAT).

Nous savons que les coëfficients des équations sont enregistrés dans la partie supérieure gauche d'une matrice de fractions ou de réels, selon que nous avons enregistré le système en mode "fraction" ou en mode "réel" (NB. La variable MODE est fixée à 0 ou 1 par la procédure DETERMINERMODE selon que toutes les lignes ont pu être enregistrées en fractions ou non).
Si nous nous trouvons en mode réel au départ, nous ne simplifierons pas la matrice.
Sinon nous réalisons la simplification en 2 parties :

- Nous simplifions d'abord les différentes fractions.
- Nous simplifions si possible, les équations qui ne contiennent que des coëfficients entiers.

Pour réaliser cela, nous avons défini une fonction PGCD . Elle reçoit 2 paramètres: le nombre d'éléments dont on cherche le PGCD et le nom du tableau où ils sont stockés. Elle renvoie le PGCD. La fonction accepte tous les nombres. Elle élimine en fait les nombres nuls et elle est basée sur un mécanisme de divisions succesives.
La procédure SIMPFRACTION parcourt succesivement tous les coëfficients de la matrice et essaie de les simplifier.
La procédure SIMPENTIER détermine les lignes qui ne contiennent que des entiers et cherche le PGCD des coëfficients de toute la ligne. Si ce PGCD est différent de 1, nous réalisons la simplification et nous vérifions si ce PGCD n'est pas "difficile à trouver" (au sens du point 2.5 page 34) . C'est le rôle de la procédure GRANDEUR qui donne le facteur restant du PGCD après en avoir extrait les puissances de 2, 3 et 5.
Selon le résultat du test entre ce facteur restant et le maximum admis (défini par le professeur), nous envoyons ou non un message qui signale qu'une simplification ardue a été réalisée.

7.2. L'ELIMINATION.

7.2.1. La procédure principale : RESOLUTION.

Notre principe initial était le suivant : nous partons de la première équation et à partir de celle-ci nous éliminons une variable dans les équations restantes. Nous partons ensuite de la 2e équation et nous éliminons une autre variable ... et ainsi de suite jusqu'à ce qu'il n'y ait plus de variables à éliminer.
Mais comme nous le verrons ce schéma de base n'est pas toujours respecté parce que - 1: Une élimination de variable n'est pas forcément nécessaire à chaque étape.

- 2: L'ordre des lignes ne sera pas toujours respecté.

Voyons d'abord dans quel ordre nous allons éliminer les variables.
Nous pourrions le faire dans un ordre arbitraire (d'abord X1, puis X2 etc...)
Nous avons choisi une méthode un peu plus intelligente : nous sélectionnons la variable qui demandera le moins d'opérations pour être éliminée, c.a.d. la variable qui apparaît dans le moins d'équations. Cela revient à sélectionner l'indice de la colonne qui contient le plus d'éléments nuls.

Exemple :
$$\begin{cases} 2X1 + 2X2 - X3 = 3 \\ X1 - X2 + X3 = 1 \\ 2X1 + X2 = 0 \end{cases}$$

Nous sélections donc X3 comme première variable à éliminer.
Comme cette variable n'apparait pas dans la 3e équation, nous ne devons évidemment pas l'en éliminer.
Nous obtenons donc le système suivant :

$$\begin{cases} 2X1 + 2X2 - X3 = 3 \\ 3X1 + X2 = 4 \\ 2X1 + X2 = 0 \end{cases}$$

Avant de passer à l'étape suivante il faut mémoriser quelle variable a déja été éliminée pour pouvoir sélectionner la variable suivante.
Une possibilité est de modifier l'ordre des colonnes : nous aurions

$$\begin{cases} -X3 + 2X1 + 2X2 = 3 \\ 3X1 + X2 = 4 \\ 2X1 + X2 = 0 \end{cases}$$

Comme nous avons déja éliminé une variable, nous ne tenons plus compte de la première colonne et nous choisissons en fonction des suivantes. Cette méthode fonctionne bien, mais elle nécessite de nombreux échanges entre les colonnes de la matrice. Plutôt que de réaliser effectivement ces échanges nous allons recourir à un vecteur d'indirection. Celui-ci contiendra l'ordre dans lequel nous

avons éliminé les variables. Ainsi au lieu d'échanger deux colonnes nous ne devons plus échanger que deux éléments d'un vecteur.

Au départ de ce vecteur d'indirection (VIND) est initialisé dans l'ordre croissant (1, 2, 3...); Ainsi, lors de la 1ère étape, on considère les colonnes dans l'ordre 1, 2, 3 ... pour effectuer la première sélection.

Reprenons l'exemple : nous sélectionnons X3. Nous changeons alors l'ordre de VIND (3, 1, 2...). A l'étape suivante, nous regardons l'ordre de prise en compte des colonnes dans VIND. Nous sommes donc arrivés au même résultat sans devoir échanger de colonnes.

Il faut également se rendre compte qu'il peut être superfflu de réaliser l'élimination d'une variable dans certains cas. En effet si à un moment donné une équation est telle qu'elle permet de déterminer la valeur d'une variable, il est plus intéressant d'injecter cette valeur dans les autres équations.

Voyons un exemple :

$$\begin{cases} 2X1 + 2X2 + X3 - X4 = 8 \\ -2X1 - X2 - X3 + X4 = 6 \\ X2 + 3X4 = 9 \\ 2X1 - 3X2 + 2X4 = 2 \end{cases}$$

Nous éliminons donc X3 :

$$\begin{cases} 2X1 + 2X2 + X3 - X4 = 8 \\ X2 = 14 \\ X2 + 3X4 = 9 \\ 2X1 - 3X2 + 2X4 = 2 \end{cases}$$

Si nous sélectionnons comme d'habitude, nous choisissons X1, nous l'éliminons et ainsi de suite. Ce n'est pas nécessaire, car l'équation 2 nous donne la valeur de X2. Si on injecte cette valeur dans les autres équations on obtient :

$$\begin{array}{l} 2X1 + X3 - X4 = -20 \\ 3X4 = -5 \\ 2X1 + 2X4 = 44 \end{array} \quad \text{avec } X2 = 14$$

Nous pouvons ainsi déterminer maintenant la valeur de X4. Injectons la dans le système et on obtient :

$$\begin{cases} 2X1 + X3 = -65/3 \\ 2X1 = 142/3 \end{cases} \quad \text{avec } X2 = 14 \text{ et } X4 = -5/3.$$

De là on tire la valeur de X1 et on obtient la solution finale.

X1 = 71/3 ; X2 = 14 ; X3 = -69 ; X4 = -5/3, avec une seule étape d'élimination.

Si nous n'avions pas détecté cette équation nous aurions continué à éliminer, alors que ce n'était plus nécessaire. Déterminer si une telle équation existe, cela revient à chercher une équation qui n'a qu'un seul coëfficient non nul. Mais il se peut également qu'elle ne comprenne aucun coëfficient non nul. Dans ce cas la suite des opérations dépend de la valeur du terme indépendant : s'il est nul, nous avons une équation du genre 0 = 0 (c.a.d. une équation indéterminée). Nous ne devons plus tenir compte de celle-ci (nous la

déplaçons en fin de système et nous incrémentons un compteur de lignes vides NBLIGVIDE afin de ne plus en tenir compte).

: s'il est non nul il s'agit d'une équation impossible et il est inutile de continuer : le système n'admet pas de solution.

Toutes ces opérations sont effectuées par la procédure YATILXVALEUR.

Il subsiste un dernier problème. Lorsqu'on sélectionne la prochaine variable à éliminer il se peut que le coëfficient de cette variable dans l'équation de base soit nul.

Exemple : $\begin{cases} X1 + X3 = 4 \\ 2X1 - X2 + 2X3 = 2 \\ 2X1 + X2 - X3 = 1 \end{cases}$

On sélectionne X2 mais pour pouvoir éliminer il faut échanger la 1ère équation avec une autre, dont le coëfficient de X2 est non nul.

Il y a également un cas limite : si une variable est absente de toutes les équations (et que bien sûr, elle n'a pas encore été "éliminée" !) nous allons sélectionner cette variable puisque la colonne correspondante ne contient pas de coëfficient non-nul. Nous allons donc essayer d'échanger la 1ère équation avec une autre pour amener un coëfficient non-nul en 1ère ligne. Ce sera évidemment impossible. Ainsi au début de chaque étape, nous vérifions si une colonne n'est pas entièrement nulle. Si c'est le cas nous placerons l'indice de cette colonne à la fin du vecteur d'indirection. La recherche du pivot se fera sur base des indices contenus dans le vecteur d'indirection, sauf pour les premières positions (qui correspondent aux variables déja éliminées) et les dernières positions (qui correspondent aux colonnes vides).

Nous verrons plus loin que cette opération facilite aussi la procédure d'interprétation des résultats.

Nous obtenons donc la procédure principale qui est décrite en annexe (voir RESOLUTION).

Nous voyons que cette procédure fonctionne avec trois paramètres (PRESENTATION, INDCHEMIN, INDGRAPH). Ceux-ci peuvent prendre chacun la valeur 0 ou 1 et ils servent à déterminer comment les résultats doivent être présentés.

Si PRESENTATION = 0 cela signifie que nous ne voulons pas présenter les résultats. Nous calculons la réponse uniquement pour pouvoir vérifier celle qui sera introduite par l'élève.

Si PRESENTATION = 1 alors les deux autres paramètres nous renseignent sur le type de présentation désiré.

Si INDCHEMIN = 1 nous suivons le cheminement, c'est à dire que nous présentons le système d'équation à chaque étape, sinon nous ne donnons que la solution finale.

Si INDGRAPH = 1 nous présentons la vue graphique pour chaque système présenté.

Nous allons maintenant décrire les principales procèdures utilisées.

7.2.2. MULTIPLICATION

Nous avons vu que c'étaient des raisons pédagogiques qui nous poussaient à travailler avec des fractions. Nous voulons également le faire le plus longtemps possible; nous essaierons toujours d'éviter le passage en réel.
Nous définissons donc l'opération de multiplication. en veillant sur deux points : - éviter unpassage inutile en réel.

- si ce passage est inévitable fournir une valeur exacte du résultat en réel .

Le résultat de la multiplication de NUM1/DEN1 par NUM2/DEN2 sera enregistré dans RESNUM/RESDEN si nous sommes toujours en mode fraction et dans RES si nous sommes en mode réel
La première opération consiste à vérifier s'il n'existe pas de facteur commun entre NUM1 et DEN2 et NUM2 et DEN1, ceci toujours dans le but de réduire les nombres traités.

Exemple : $\frac{1000}{9} \times \frac{63}{1000}$

- En effectuant directement les deux multiplications nous aurions 63.000/9000 et il y aurait dépassement de capacité.
- Nous éliminons le facteur 1000 qui est le PGCD entre NUM1 et DEN2 et nous obtenons $\frac{1}{9} \times \frac{63}{1} = \frac{63}{9}$. Il ne reste plus qu'à simplifier.

7.2.3. SOUSTRACTION.

Notre souci est bien sûr le même. C'est pour cela que nous cherchons le PGCD des 2 dénominateurs avant d'effectuer l'opération.

Exemple : $\frac{47}{400} - \frac{21}{1000}$

- En effectuant directement on obtient $\frac{(47 \times 1000) - (21 \times 400)}{(400 \times 100)}$ ce qui provoque évidemment des dépassements.
- Nous avons réduit les 2 dénominateurs d'un facteur 200 et nous obtenons $(\frac{47}{2} - \frac{21}{5}) \times \frac{1}{200}$ et nous obtenons la solution en fraction.

7.2.4. La procédure COLVIDE .

C'est cette procédure qui vérifie s'il n'y a pas de colonne vide dans la partie de matrice où l'on va chercher quelle variable il faut éliminer .
Cette partie de matrice comprend les lignes comprises entre POSLIGNE (qui est la position de la ligne courante de base pour l'élimination) et NBLIGNE - NBLIGVIDE. Nous ne considérons pas les premières lignes :

- Celles qui sont de la forme Xi = C et qui donnent la valeur d'une variable.
- Celles qui ont déja servi de base à l'élimination.

ni les dernières (qui sont du type 0 = 0).
Elle comprend les colonnes dont le numéro est repris dans la partie du vecteur d'indirection comprise entre (NBXELIM + 1) et (INDMAX - NCOLVIDE). Nous ne tenons donc pas compte des colonnes correspondant à une variable déjà éliminée, ni des colonnes vides.
Les résultats sont:- un booléen VIDE vrai si une colonne dans la zone décrite est entièrement nulle.

- un entier INDICE qui, si vide est vrai, contient le numéro de la colonne vide.
- un entier POSINDICE qui donne la position du numéro de la colonne vide dans VIND. Cela facilite l'échange dans VIND.

7.2.5. Les procédures FININD et DEBUTIND.

Ces deux procédures servent à modifier l'ordre du vecteur d'indirection.
FININD est utilisé pour amener un numéro de colonne vide vers la fin de VIND (plus précisément à la première place des colonnes vides).
Les autres éléments du vecteur, qui étaient à droite de ce numéro, sont décalés d'un rang vers la gauche.
DEBUTIND est utilisé au contraire pour amener un numéro de colonne vers le début de VIND (à la dernière place des variables déjà avancées par DEBUTIND). soit parce que la variable correpondante a une valeur bien déterminée, soit parce qu'elle vient d'être éliminée.
Les autres éléments du vecteur, qui étaient à gauche de ce numéro, sont déplacés d'un rang vers la droite.

7.2.6. La procédure YATILXVALEUR.

Cette procédure travaille également sur la zone de la matrice décrite en 4.2.3.

Elle vérifie s'il n'y a pas une équation:

- qui permette de donner une valeur à une variable(c.a.d. une équation avec un seul coëfficient non nul.)
- impossible, c.a.d. du type 0 = C (avec C ≠ 0)
- indéterminée, c.a.d. du type 0 = 0.

Cela revient donc à compter pour chaque ligne le nombre de coëfficients nuls. Si ce nombre est 0, selon la valeur du terme indépendant (= ou ≠ 0) nous déplaçons cette équation dans la zone des lignes vides ou nous positionnons l'indicateur IMPOSSIBLE à vrai pour arrêter. Si ce nombre est 1 cela signifie qu'une variable a une valeur bien déterminée : nous positionnons TROUVE et LIGNE.

7.2.7. La procédure DONNERVALEUR.

Nous venons de trouver une équation dont un seul coëfficient est non-nul.
Nous allons déterminer la valeur de la variable correspondante et l'enregistrer dans un tableau de solutions (XF : pour les fractions , XR : pour les réels).
Chaque élément du tableau XF est composé de trois parties: un numérateur et dénominateur, ainsi qu'un booléen VALEUR qui est vrai si la variable d'indice correpondant a une valeur déterminée.
Chaque élément de XR est composé de 2 parties : la solution en réel et le booléen VALEUR.
Si nous sommes en mode 0 l'équation a la forme :

$$(\text{INTER}[\text{LIGNE},\text{INDICE}].\text{NUM}/\text{INTER}[\text{LIGNE},\text{INDICE}].\text{DEN})*X[\text{INDICE}] = \frac{\text{INTER}[\text{LIGNE},\text{INDMAX}].\text{NUM}}{\text{INTER}[\text{LIGNE},\text{INDMAX}].\text{DEN}}$$

Déterminer la valeur de ce "X[INDICE]" c'est multiplier la 2e fraction par l'inverse de l'autre. Remarquons que si le coëfficient est négatif cela signifie que son numérateur est négatif, car nous avons pris comme convention de placer le signe devant toute la fraction. Comme nous allons multiplier par l'inverse de la fraction, le numérateur va devenir un facteur du dénominateur du résultat et ne peut donc être négatif pour respecter cette convention. Comme inversément le dénominateur va devenir un facteur de numérateur il peut être négatif : nous inversons donc les signes.
Selon que le résultat dépasse ou non la capacité des entiers nous enregistrons le résultat dans XF ou XR.
En mode 1 l'équation est : (INTREEL[LIGNE, INDICE])*X[INDICE] = INTREEL [LIGNE, INDICE]: il suffit d'effectuer la division et d'enregistrer le résultat.

7.2.8. CHANGERORDREDESLIGNES.

Comme nous venons de déterminer la valeur d'une inconnue, nous avons une équation de la forme X INDICE = terme indépendant.
Cette équation, nous devons la retirer de la zone de la matrice dans laquelle nous effectuons nos tests. Nous l'amenons en première position et nous décalons les autres d'un rang.

7.2.9. INJECTER.

Nous allons retirer la variable dont nous venons de trouver la valeur, en la remplaçant par celle-ci dans toutes les autres équations.
En mode fraction, il y a une multiplication et une soustraction à effectuer, ce qui peut bien sûr entrainer un passage en réel.

7.2.10 CHERCHERPIVOT.

Cette procédure cherche le pivot suivant, c.a.d. l'élément à partir duquel pourra s'effectuer l'élimination suivante. Cela s'effectue en deux étapes :

- chercher la variable qui apparaît le moins dans la zone considérée (voir point 4.2.3.)
- vérifier si la ligne de base pour l'élimination contient bien cette variable. Sinon, nous échangeons cette ligne avec la première ligne contenant un coëfficient non-nul pour la variable.

Nous positionnons également un booléen, s'il n'y a pas d'élimination à effectuer (parce que la variable n'apparaît que dans une seule équation).

7.2.11. ELIMPIVOT.

Cette procédure réalise l'élimination d'une variable de toutes les équations de la zone considérée, sauf la première.
Revoyons sur un exemple le principe même de l'élimination.

$$\begin{cases} 3X1 - 2X2 + X3 = 5 \\ 2X1 + X2 + 2X3 = 12 \\ X1 - X2 - X3 = 0 \end{cases}$$

Pour éliminer X^1 des 2 dernières équations, il faut :

1. multiplier la 1ère équation par le coëfficient de X1 dans la 2e, multiplier la 2e équation par le coëfficient de X1 dans la 1ère, soustraire et obtenir ainsi la nouvelle valeur de la seconde équation.

$$2 \times (3X1 - 2X2 + X3 = 5)$$
$$-3 \times (2X1 + X2 + 2X3 = 12)$$
$$0X1 - 7X2 - 4X3 = -26$$

2. Même chose avec la 3e équation.

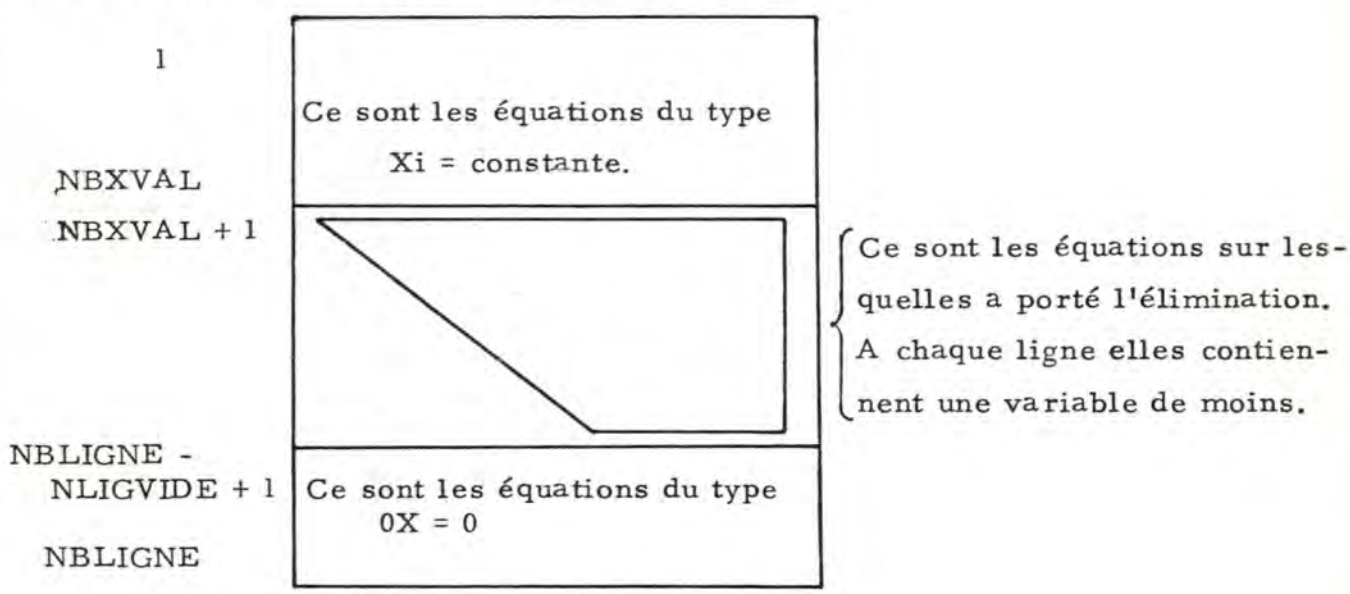
Nous stockons donc la valeur du coëfficient de la variable dans l'équation de base pour toute la durée de l'élimination(dans MUL2 ou MULR2 suivant le mode.) Ensuite pour chaque ligne, nous enregistrons la valeur du coëfficient de la variable dans cette ligne. Nous multiplions l'équation de base par cette variable et cette équation par la valeur du pivot. Nous effectuons ensuite la soustraction.

Comme toujours, si nous sommes en mode fraction, une opération de multiplication ou de soutraction risque de nous faire passer en réel.

7.3. INTERPRETATION.

7.3.1. La procédure principale INTERPRETATION.

Cette troisième partie de la résolution ne s'effectue que si, pendant l'élimination, nous ne nous sommes pas aperçus que le système était impossible.
Nous pouvons schématiser la matrice en 3 parties :



Nous allons rechercher si, dans la deuxième partie, nous ne pouvons pas trouver une équation qui permette de déterminer la valeur d'une variable.
cette procédure sera semblable à YATILXVALEUR (voir point 7.2.6), mais comme l'élimination a été réalisée de haut en bas, ce sont les lignes inférieures qui contiennent le moins de variables. Cette procédure prendra donc les équations en compte de bas en haut. Elle détectera également les équations impossibles et indéterminées. Comme dans la phase d'élimination, si nous trouvons la valeur d'une inconnue, nous l'injectons dans les autres équations.

Lorsqu'il n'y a plus de telles équations, il y a deux possibilités:

1. La deuxième partie de la matrice est vide; cela signifie que toutes les variables ont une valeur bien précise et que le système admet une solution unique.
2. Il reste une ou plusieurs équations dans cette partie et aucune d'elles ne permet de trouver la valeur d'une inconnue ; il est donc impossible d'attribuer une valeur à chaque inconnue et le système est indéterminé. C'est le cas si le nombre d'inconnues est supérieur au nombre d'équations "réelles". Par ce terme, nous entendons les équations qui imposent une nouvelle contrainte (c.a.d. qu'elles ne sont pas combinaisons linéaires d'équations précédentes). Or les équations qui sont ainsi combinaisons linéaires conduisent lors de l'élimination à des lignes vides (0 = 0). L'ordre d'indétermination est donc égal au nombre d'inconnues, moins le nombre d'équations "réelles", c.a.d. INDMAX - (NBLIGNE - NLIGVIDE).
   Nous pouvons choisir arbitrairement quelles variables sont indéterminées (les seules qui ne peuvent pas l'être sont celles qui ont une valeur déterminée, c.a.d. celles qui sont au début du vecteur d'indirection).
   Nous choisissons celles qui sont les dernières dans VIND, c.a.d. celles qui n'ont pas été éliminées ou celles dont la colonne de coëfficients était entièrement nulle à un moment donné (voir procédure COLVIDE en 4.2.3). Nous mémorisons ces variables dans un vecteur QCQ qui sera utilisé pour exprimer la solution indéterminée.

Décrivons maintenant les procédures nouvelles nécessaires à l'interprétation.

7.3.2. XVALEUR.

Cette procédure a exactement les mêmes buts que la procédure YATILVALEUR (voir point 7.2.5). Pourquoi dès lors en créer une autre ?
Il y a 2 raisons :

1. Comme nous l'avons dit plus haut nous prenons les lignes en considération dans l'autre sens, car c'est dans les dernières équations que nous avons le plus de chances de pouvoir déterminer la valeur d'une inconnue.
2. Il ne faut plus tenir compte du vecteur d'indirection pour prendre les colonnes en considération, car au fur et à mesure que l'on injecte des valeurs dans le système, nous devrons également rechercher si les variables qui ont déja été éliminées ne peuvent être déterminées.

7.3.3. SOLUNIQUE.

Cette procédure affiche la valeur de chaque inconnue à l'écran.

7.3.4. SOLINDETERMINEE.

Cette procédure a pour but d'exprimer une solution indéterminée, c.a.d. de donner la valeur des variables déterminées et d'examiner les relations existant avec les autres variables.
Nous affichons le type de la solution, la valeur des variables déterminées et les "X" que nous avons choisis comme indéterminés (ceux qui sont dans le vecteur QCQ).
Il reste alors, à partir des équations situées dans la deuxième partie de la matrice, à exprimer les "X" restants en fonction des indéterminés et des autres "X". Pour ce faire, nous utilisons un vecteur VU, qui à un moment donné, contient les indices des variables qui ont déjà été traitées (soit qu'elles sont indéterminées, soit qu'elles ont déjà été exprimées) et une fonction DEJAVU qui est vraie si la variable a déjà été traitée (c.a.d. si elle appartient à VU).
Nous traitons les équations de cette 2e partie de bas en haut. Nous exprimons la valeur de la première variable non encore traitée de la ligne grâce à la procédure EXPRIMERX. Celle-ci fait elle-même appel à la procédure DENOM, qui transforme une fraction donnée en une chaîne de caractères et qui ne reprend le dénominateur que s'il est différent de 1.

8. VERIFICATIONS DES REPONSES.

Comme nous l'avons déjà répété maintes fois, cette partie n'en est qu'au stade embryonnaire. D'après le scénario, nous voyons qu'elle se situe entre le moment où l'on demande à l'utilisateur quel est le type de sa solution et celui où l'on oriente le programme selon que la réponse est correcte ou non.
Son rôle est donc de saisir les réponses, de les comparer et de déterminer leur validité. Eventuellement en cas d'erreur nous envoyons un message qui signale le "genre" de l'erreur(voir AFFICHAGE, INIER 1 à 5).
Ainsi, nous demandons à l'utilisateur si sa solution est unique, si le système est indéterminé ou impossible.
Dans le premier cas (voir VSUNIQUE), nous vérifions d'abord si la solution est bien unique. Si ce n'est pas le cas nous positionnons CORRECT à faux et nous signalons l'erreur. Sinon nous demandons à l'utilisateur d'introduire sa solution (et éventuellement de la corriger). Si la solution calculée peut s'exprimer en fractions, celles-ci sont simplifiées. Nous appelons donc la procédure SIMPLISUNIQUE qui détermine si la solution introduite est en mode fraction ou en mode réel, et qui, dans le premier cas réduit les fractions si nécessaire.
Nous comparons ensuite les 2 modes d'enregistrement (nous estimons que si une solution est en réel et l'autre en fraction, il y a erreur). Si les modes sont les mêmes nous comparons simplement les valeurs une à une.
Notons que dans le cas des fractions, nous comparons bien sûr les fractions simplifiées. (voir COMPFRACTION et COMPREEL).
Dans le deuxième cas (voir VSINDETERMINEE), nous vérifions si le système est bien indéterminé. Si non, nous positionnons CORRECT à faux et nous envoyons un message d'erreur. Si oui, nous demandons d'introduire l'ordre d'indétermination. Nous vérifions celui-ci et selon le résultat nous positionnons CORRECT.
Dans le troisième cas (voir VSIMPOSSIBLE), nous vérifions uniquement s'il s'agit réellement d'un système impossible.
Si oui,nous considérons la réponse correcte. Si non, il y a erreur et nous le signalons.

9. CREATION D'EXERCICES.

La procédure en oeuvre actuellement est arbitrairement simple. Nous utilisons la fonction RANDOM (disponible dans APPLESTUFF) qui renvoie un nombre entier aléatoire (compris entre 0 et 32767).

Nous avons pris comme option de créer des systémes d'équations où les coëfficients sont tous des entiers compris entre 0 et 5. Ainsi nous demandons le nombre d'équations et d'inconnues souhaitées et nous garnissons les différents éléments de la matrice MAT. Chaque numérateur sera un entier compris entre 0 et 5 (RANDOM MOD 6). Chaque dénominateur sera 1 puisque nous voulons des coëfficients entiers.

10. DEMO 2 x 2.

Comme nous l'avons expliqué au point 2.8, ce module compte 2 parties:

1. Nous présentons d'abord quelques droites. Nous expliquons que si un des coëfficients A ou B est nul, la droite correspondante est parallèle (ou confondue) à un des axes.
   Ensuite nous montrons que si aucun coëfficient n'est nul, la droite a un point d'intersection avec chaque axe.
   Enfin nous donnons un exemple de droite passant par l'origine.

2. Nous présentons les 3 types possibles de systèmes. Nous montrons que :
   - si le système admet une solution unique, celle-ci est située à l'intersection des 2 droites.
   - si le système est indéterminé, les 2 droites sont confondues.
   - si le système est impossible, les 2 droites sont parallèles.

Remarque : Afin de présenter les différents textes explicatifs nous avons défini une procédure en tous points similaires à AFF TEXTE (voir AFF TEXT 2).

11. DEMO 3 x 3.

La présentation de cette démonstration s'effectue en 3 parties :

1. Nous présentons d'abord quelques plans, afin de familiariser l'utilisateur avec les représentations.
   Nous lui montrons d'abord comment nous représentons un plan où aucun coëfficient n'est nul en expliquant quels sont les points qui nous servent de points de repère. Puis nous montrons ce qui se passe si un des coëfficients est négatif (avec éventuellement changement de la position de l'axe X2, figure "doublée" ...).
   Nous proposons ensuite la manière choisie pour représenter un plan passant par l'origine.
   Enfin nous terminons par plusieurs exemples où certains coëfficients A, B et/ou C sont nuls, afin de montrer leurs particularités.

2. Nous présentons ensuite les 3 possibilités existant pour la superposition de 2 plans.
   Nous montrons un exemple de plans confondus et de plans parallèles pour montrer qu'il existe respectivement dans ces cas des combinaisons linéaires sur 4 ou 3 coëfficients.
   Nous expliquons ensuite ce que nous montrons en règle générale de la droite d'intersection entre 2 plans sécants. (Pour rappel, il s'agit du segment qui lie le point d'intersection sur le plan horizontal à celui sur le plan de face).
   Nous montrons alors sur 3 exemples qu'il est parfois nécessaire de "prolonger" les plans pour pouvoir visualiser les 2 points.
   Nous terminons par montrer quelques cas un peu plus particuliers (par exemple: la droite d'intersection est parallèle à un des plans de référence, les 2 plans passent par l'origine ...).

3. Nous montrons enfin les différentes possibilités qui existent pour un système 3 x 3.
   Nous montrons 2 exemples où la solution est unique. Nous citons les différentes situations qui peuvent conduire à des systèmes impossibles et celles qui amènent à des systèmes indéterminés d'ordre 1 ou 2.

Nous faisons également la même remarque que pour DEMO 2 en ce qui concerne le texte d'explication.

12. COORDINATEUR.

Sa fonction est d'organiser le déroulement du programme comme il est décrit dans le scénario.
Nous avons vu que toute la structure est basée sur le menu principal qui propose les 5 choix de départ. Après chaque exécution d'une séquence spécifique, l'utilisateur a la possibilité de revenir à ce menu principal. Le programme est donc une boucle qui affiche ce menu, et selon le choix de l'utilisateur, appelle une procédure spécifique. La boucle se termine lorsque l'indicateur FIN est vrai (celui-ci est positionné à vrai lorsque l'utilisateur demande à quitter le programme).
Nous allons maintenant décrire ce qui se passe pour chacun des 5 choix possibles.

12.1. RESPROBLEME.

L'utilisateur demande que l'ordinateur résolve un problème qu'il va introduire. Nous savons qu'après une première résolution, l'utilisateur pourra changer d'exercice, repartir du même exercice et le corriger ou retourner au menu-principal.
Donc, sauf s'il choisit le retour au menu principal, nous resterons dans la résolution du problème.
La structure RESPROBLEME est donc également une boucle (qui s'exécute jusqu'à ce que RETOURMENU soit vrai). Dans cette boucle nous demandons d'introduire le système ou de le corriger, selon que l'utilisateur veut introduire un nouveau système ou qu'il désire modifier le précédent. Ensuite nous proposons les 4 possibilités de solution et selon le choix nous lui la présentons. (voir CHOIXSOLUTION). Et enfin nous demandons ce qu'il souhaite pour continuer.
Pour cela nous avons utilisé la procédure INTROSYSTEME. Celle-ci demande si l'utilisateur désire ou non les instructions et le cas échéant les lui présente. Elle fait appel à la routine d'introduction et à celle de correction du système (CORSYS).
Nous avons aussi appelé la procédure CORRECTION qui permet de modifier un système déjà introduit. Elle utilise également CORSYS.

12.2. VERIFACTIONPROBLEME.

L'utilisateur demande que l'ordinateur vérifie sa réponse à un problème qu'il va introduire ou qui sera créé par l'ordinateur.
De nouveau cette procédure a une structure de boucle. Celle-ci s'exécute jusqu'à

ce que l'on veuille revenir au menu principal.
Dans cette boucle, à moins que l'utilisateur ne désire recommencer le même exercice, nous lui demandons s'il veut introduire un exercice ou s'il veut que l'ordinateur en crée un.
Ensuite nous acceptons sa solution (voir ACCEPTERSOLUTION) et nous la vérifions (VSUNIQUE, VSINDETERMINE, VSIMPOSSIBLE) ce qui a pour effet de positionner CORRECT à vrai ou faux.
Selon la valeur de CORRECT, nous effectuons SOLCORRECT ou SOLINCORRECT.
Ces deux procédures proposent les différents choix prévus dans les deux cas et ont également une structure de boucle.

12.3. DEMO 3 et 12.4 DEMO 2

Dans les 2 cas, il s'agit d'un simple appel à cette procédure.

12.5. FINPROGRAMME.

L'utilisateur demande à quitter le programme. Nous positionnons donc FIN à vrai, ce qui fera sortir de la boucle principale et donc entraînera la fin du programme.
Nous présentons un dernier écran (de séparation avec l'utilisateur).

13. PROBLEMES RENCONTRES LORS DE L'INTEGRATION.

Avant d'en arriver au coeur de ces problèmes, il faut commencer par parler du fonctionnement du micro-ordinateur APPLE II.

Pour exécuter un programme de grosse taille, il est nécessaire de le découper en différentes parties qui peuvent chacune "tenir" en mémoire centrale. Deux possibilités sont offertes par le PASCAL "UCSD" disponible sur l'APPLE.

- Déclarer quelques grosses procédures (ou fonctions) comme "segments". Cela a pour effet de ne charger ces procedures et fonctions en mémoire centrale que l'orsqu'elles sont appelées effectivement.
- Diviser le programme en différentes parties appelées "UNIT". Chaque "UNIT" constitue une unité compilable séparément. En donnant une directive appropriée au compilateur (la directive "NOLOAD"), il est également possible de ne charger chaque "UNIT" que lorsque l'on en a besoin.

Nous avons choisi d'utiliser des UNITS. En effet, utiliser des segments imposait d'avoir un gros programme unique, car à l'intérieur des UNITS il n'est pas permis de déclarer des segments. Donc toute compilation prendrait un temps énorme. De plus le nombre de segments que l'on peut déclarer est limité et ceux-ci doivent obligatoirement se trouver en tête du programme. D'autre part,l'emploi des UNITS est assez avantageux car, une fois la UNIT définie, on peut l'utiliser dans n'importe quel programme. A titre d'exemple, citons le cas de "TURTLEGRAPHICS" qui est une UNIT prédéfinie permettant de réaliser des graphiques. Pour pouvoir utiliser ces procédures et fonctions il suffit de preciser au début du programme que l'on va utiliser TURTLEGRAPHICS. (Par l'ordre "USES TURTLEGRAPHICS").

Utiliser les UNITS impose évidemment une structure spéciale. Chaque UNIT a un nom, une partie "INTERFACE" et une partie "IMPLEMENTATION".
La partie "INTERFACE" contient la partie de la UNIT qui est "visible de l'extérieur", c.a.d. - Les éventuels appels à d'autres UNITS.

- Les éventuelles déclarations de données (celles-ci sont alors communes à la UNIT et aux programmes qui l'utilisent).
- La ou les procédures (ou fonctions) qui sont "appelables" par les programmes qui utilisent cette UNIT.

La partie "IMPLEMENTATION" comprend le texte des procédures. Il n'y a pas de conventions spéciales, si ce n'est qu'il ne faut pas reprendre les para-

mètres des fonctions et procédures déja déclarées en "INTERFACE".

Nous avons développé un programme principal (COORDI-(NATEUR)) qui utilise une dizaine de UNITS.

- "APPLESTUFF" et "TURTLEGRAPHICS" qui sont prédéfinies; la 1ère nous permet de générer des sons et des nombres aléatoires et la seconde la réalisation des graphiques.
- "UTILUNIT" elle comprend d'une part la déclaration de certaines données qui seront pratiquement globales parce que presque toutes les UNITS utiliseront celles-ci. On y retrouve la déclaration des types "FRACTION", "MATRICE" .... D'autre part elle comprend un ensemble de procédures utilitaires tels impression de matrices, blocage et déblocage du défilement de l'écran, émission de "BIP" sonore ...
- "INTROUNIT": Cette UNIT regroupe toutes les procédures nécessaires à l'introduction des systèmes d'équations et des réponses.
- "SYSUNIT" : Elle permet la résolution des systèmes.
- "AFFICHAGE" : Elle permet de présenter des textes, des messages et des menus à l'écran.
- "DESSIN22" : Elle permet de représenter graphiquement les systèmes 2 x 2.
- "DECLAR33" : Dans cette UNIT nous déclarons les structures de données nécessaires à la représentation graphique des systèmes 3x3. Nous procédons ainsi pour avoir une structure de données commune aux différentes UNITS qui réalisent cette représentation ("CHOIXPLAN", "INTER33", "DESSIN33").
- "CHOIXPLAN" : Elle détermine par quels sommets nous allons représenter un plan et elle étudie les orientations permises.
- "INTER33" : Elle détermine si 2 plans sont confondus, parallèles ou sécants et dans ce dernier cas détermine la position de la droite d'intersection (avec éventuels prolongements de plans).
- "DESSIN33" : Elle fait notamment appel aux 3 UNITS précédentes et permet la représentation des plans déterminés à l'écran.

Nous devons essayer de "faire tourner" le programme ainsi défini sur l'APPLE II. Celui-ci à une mémoire de 48 K, mais une fois le système PASCAL chargé il ne reste déjà plus que 18828 mots-mémoire disponibles.

Il est bien certain que ces différentes UNITS ne sauront être chargées en mémoire en même temps. Nous allons donc utiliser l'option "NOLOAD". Il faut savoir qu'ainsi le corps de la UNIT (partie "IMPLEMENTATION") n'est chargée qu'en cas de besoin. Mais la partie "INTERFACE" est chargée de toute façon. C'est à partir de là que commencent nos problèmes .

En effet en compilant le programme principal (les "USES" de toutes les UNITS doivent y figurer), nous avons rencontré un premier problème : il n'y avait pas suffisament de place pour le compiler. Nous avons toutefois pu le résoudre en modifiant la structure des interfaces (déclarer plus de données locales, déclarer moins de procédures en "INTERFACE", ...).
Mais après cela en essayant d'exécuter il manquait encore de la place.
En effectuant des mesures sur la taille de la mémoire encore disponible après chargements des différentes UNITS (et ce avec ou sans option "NOLOAD" pour déterminer la partie mémoire nécessaire à l'interface et à l'implémentation), nous sommes arrivés aux résultats suivants :
Toutes les places sont exprimées en mots-mémoire encore disponibles.

Place disponible au départ : + ou - 18800 mots.

UNIT	INTERFACE	CHARGEMENT COMPLET
APPLESTUFF	4	342
TURTLEGRAPHIC	6271	9305
AFFICHAGE	1	5003
UTILUNIT	2193	4657
SYSUNIT	1	3966
INTROUNIT	2051	4772
DESSIN22	69	3344
DECLAR33	298	316
CHOIXPLAN	2	4912
INTER33	2	5592
DESSIN33	1	2465

Au total pour charger toutes les interfaces nous avons besoin de + ou - 10900 mots (dont 6300 rien que pour "TURTLEGRAPHICS"). Si en plus nous chargeons le programme principal, il nous reste en tout et pour tout 5900 mots. Cela entraine que si à un quelconque moment, nous voulons utiliser une procédure de TURTLEGRAPHICS, cette UNIT doit être chargée et il faut encore compter plus de 3000 mots supplémentaires, cela implique qu'il reste moins de 2900 mots disponibles.

Vu la taille de nos UNITS , il nous a donc été impossible de réaliser l'intégration complète prévue au départ car , bien évidemment nous ne nous attendions pas à ce que "TURTLEGRAPHICS" soit une UNIT aussi gourmande. A elle seule elle occupe la moitié de la mémoire disponible. Une solution radicale serait, si on peut avoir accès au programme source de "TURTLEGRAPHICS" d'en éliminer tout ce que nous n'utilisons pas afin de réduire la taille ou encore de se créer sa propre UNIT "GRAPHIQUE" avec les quelques procédures nécessaires.

Sinon quelles seraient les directions à suivre pour tenter de résoudre ce problème ?

- Premièrement modifier la structure de la UNIT de résolution "SYSUNIT" de telle manière à ne plus appeler les procédures graphiques à partir d'elle, mais bien du COORDINATEUR. Mais cela ne suffira pas parce que même ainsi nous ne pourrions exécuter.

- On se rend compte que 2 de nos UNITS ont une interface importante. Il s'agit de:"UTILUNIT" Il sera difficile d'y gagner de la place car elle comprend les déclarations globales (de types comme "FRACTION", "MATRICE" ... et de variables comme (MAT, INTER, MREEL, INTREEL....). De plus les procédures et fonctions qui y sont déclarées peuvent être appelées de partout.

  :"INTROUNIT" Ici il est possible de regagner de la place en déclarant les matrices de transistions et d'actions locales (leur initialisation devra être également locale).

- On devrait également découper plus finement les différentes UNITS qui utilisent "TURTLEGRAPHICS" pour que la partie "IMPLEMENTATION" ne dépasse pas 2000 mots.

Comme nous pouvons le constater ce problème ne sera pas résolu simplement car premièrement ce sera très long et deuxièmement le faible espace disponible après chargement "TURTLEGRAPHICS" impliquera certainement une simplification des mécanismes décrits.

14. CONCLUSION ET AMELIORATIONS POSSIBLES.

Nous concluons ce travail en formulant deux souhaits :

1. Nous espérons que ce programme sera effectivement utilisé comme support à l'enseignement des systèmes d'équations linéaires. D'une part cela permettrait de voir à quel point les objectifs recherchés sont atteints. En particulier il serait intéressant de savoir si la compréhension de l'élève en est modifiée.

   D'autre part, une utilisation sur le terrain mettrait sans doute certaines lacunes à jour et indiquerait quelles modifications ou améliorations seraient souhaitables.

2. Nous espérons également que ce travail connaîtra des prolongements, nous l'avons déjà souvent répété.
   D'une part, il y a certaines améliorations et certains changements que nous avons envisagés dès à présent (nous les citons ci-dessous) et d'autre part l'utilisation devrait en faire découvrir d'autres.
   De toute façon, dans les éventuelles versions futures, les outils de base décrits dans ce travail pourront être réutilisés.

Nous allons maintenant détailler différents points auxquels nous avons songé. Nous les classons en 2 catégories : - l'amélioration de ce qui existe.
- la définition de modules supplémentaires.

(Cette liste pourrait être allongée indéfiniment mais nous nous limitons à présenter quelques idées).

14. 1. AMELIORATION DE CE QUI EXISTE.

14. 1. 1. Vérification des exercices.

Nous l'avons signalé à plusieurs reprises : la méthode adoptée est nettement insuffisante du point de vue pédagogique et elle ne devrait normalement constituer qu'une étape. De plus elle n'est pas complète puisqu'en cas de solution indéterminée, nous nous contentons de vérifier l'ordre d'indétermination. Une première amélioration, relativement simple, consisterait à permettre à l'utilisateur d'introduire cette solution indéterminée et de la vérifier. Cela entraînerait peu de changement car les outils sont prévus pour cela. Par exemple : l'expression de solution indéterminée dépend du choix des variables indéterminées. Ce choix est arbitraire de la part de l'élève mais cela ne posera pas de problème. Pour exprimer la solution de ces variables il suffira d'assigner la valeur de

leurs indices à un vecteur de nom QCQ, qui sert de base à cette expresssion.
Bien sûr, il serait plus intéressant du point de vue pédagogique de pouvoir suivre pas à pas la démarche de l'élève. Cela implique qu'à chaque étape nous saisissions le système d'équations modifié.
Cela pourra se faire sans problème grâce au module d'introduction. Il faudra ensuite vérifier si la modification apportée au système est correcte. Pour cela il faudra réaliser la même élimination que l'élève . Cela ne pose pas non plus de problème puisque nous disposons d'une procédure qui permet d'éliminer sur un pivot donné(ELIMPIVOT). Il serait alors assez simple de "situer l'erreur". (Par exemple: signaler pour quelle(s) équation(s) et quelle(s) inconnue(s) le coëfficient est erroné).
Par contre si nous voulons "comprendre" l'erreur et l'expliquer (par exemple : erreur dans la soustraction de 2 équations , dans la multiplication d'une équation par 1 coëfficient ...), il n'y a pratiquement pas de limite quant à la complexité de l'outil à définir.

14.1.2. Introduction des systèmes et des réponses.

Des améliorations pourraient être apportées à l'effacement des caractères. Le système que nous avons adopté est très fruste (effacement de toute la ligne et uniquement pour la ligne en cours). On pourrait imaginer un système proche de certains éditeurs de texte, basé sur la position d'un curseur que l'on peut déplacer de ligne en ligne et de caractère en caractère et où sont prévues des fonctions d'effacement, d'insertion , de remplacement ...
Notre but n'était évidemment pas de réaliser un éditeur perfectionné, mais de toute façon le problème qui se pose à l'effacement d'un caractère est qu'il faut pouvoir revenir à le même situation qu'avant son enregistrement, c.a.d. qu'il faut être capable de "défaire" les actions réalisées pour l'enregistrement de ce caractère.

14.1.3. Représentations graphiques.

Nous avons cité (points 1.1.1 et 2.2.2) quelques cas où les représentations graphiques pouvaient laisser à désirer. Il est certain que nous aimerions voir ces problèmes résolus, même si pour le moment nous n'avons pas encore réfléchi à la manière de le faire.
En présentant les différents types d'intersection entre deux plans nous avons relevé l'intéret de recourir à des traits pleins et pointillés (voir point 2.2.2). Remarquons que cette possibilité est prévue dans la procédure DROITE où l'indicateur INDPOINTILLE peut être ou non positionné. Il est clair que la

difficulté n'est pas de complèter cette procédure pour permettre de représenter les pointillés mais bien de modifier la procédure qui trace les plans, car il faudra déterminer quelles portions de droites doivent être représentées en pointillés et en continu .

14.1.4. Création d'exercices.

Comme nous l'avons vu(point 2.7 + chap. 9) cette routine est excessivement simple. Il est bien sûr qu'un nombre illimité de variantes est imaginable. Chacun peut évidemment avoir son idée sur le sujet et cette routine pourra éventuellement devenir très complexe.

14.2. NOUVEAUX MODULES ENVISAGES.

14.2.1. Module d'initialisation.

Ce serait un module qui permettrait au professeur de modifier, chaque fois qu'il le désire,des paramètres enregistrés dans un fichier.
Ceux-ci concerneraient les conditions de fonctionnement du logiciel.
Nous envisageons cela comme un moyen de :

- Paramètrer les menus (voir exemple au point 1.2
  En enregistrant le nombre de choix pour chaque menu, il serait possible de restreindre les choix offerts à l'élève. Cela procurerait une plus grande souplesse et permettrait de modifier les scénarios possibles en fonction du niveau des élèves.
- Permettre au professeur d'introduire (et modifier) lui-même les exemples qu'il souhaite voir représentés dans les modules de démonstration.
- Permettre au professeur de modifier les couleurs définies pour les représentations graphiques, de définir son "MAXADMIS" (voir point 2.5) ....

14.2.2. Introduction aux équations avec un paramètre.

Nous pourrions montrer aux élèves, par exemple, ce que représente une équation à 3 inconnues où un des coëfficients est paramètrique. Une représentation graphique pourrait être obtenue simplement en donnant successivement à ce paramètre différentes valeurs. En superposant ces différents graphiques nous obtiendrions une animation à l'écran qui montrerait comment "bouge" un plan avec un paramètre.

PROJET D'ENSEIGNEMENT ASSISTE PAR ORDINATEUR:

"LES SYSTEMES D'EQUATIONS LINEAIRES."

A N N E X E S .

Facultés Universitaires
Notre-Dame de la Paix
Institut d'informatique

PROJET D'ENSEIGNEMENT ASSISTÉ PAR ORDINATEUR : LES SYSTÈMES D'ÉQUATIONS LINÉAIRES.

Année académique
1983-1984

Mémoire présenté
par Yves Jenné
en vue de l'obtention
du grade de licencié
et maître en informatique.

Promoteur : Cl. Cherton.

ANNEXES

ELLES COMPRENNENT :

* UNE LISTE ALPHABETIQUE DES PROCEDURES (AVEC LEURS REFERENCES).
  (voir pp 1-4).

* LE TEXTE DES DIFFERENTES "UNITS" QUE NOUS AVONS CREEES.
  (voir pp 6-41 pour les "units" generales).
  (voir pp 43-81 pour les "units" qui servent aux representations graphiques).
  (voir pp 83-95 pour les "units" qui servent a l'affichage a l'ecran).

* LE TEXTE DES DIFFERENTS PROGRAMMES.
  (voir pp 97-117).

LISTE ALPHABETIQUE DES PROCEDURES.

TRUNPLAN	DEMO3		111
TRUNPLAN	GRAPH		115
UNEDROITE	DEMO2		108
UNEDROITE	GRAPH		116
UNPLAN	DESSIN33		78
VALIDATION	AFFICHAGE		86
VERIFICATIONPROBLEME	COORDI		101
VSIMPOSSIBLE	COORDI		100
VSINDETERMINEE	COORDI		99
VSUNIQUE	COORDI		98
XVALEUR	SYSUNIT		36
YATIXVALEUR	SYSUNIT		31

```
**************************************
*                                    *
*  U N I T S   G E N E R A L E S  *
*                                    *
**************************************
```

* UTILUNIT : ELLE COMPREND
  - LES DECLARATIONS GLOBALES
  - DES PROCEDURES ET FONCTIONS UTILITAIRES

(voir pp 6-16)

* INTROUNIT : ELLE COMPREND LES PROCEDURES UTILISEES A L'INTRODUCTION DES SYSTEMES ET DES SOLUTIONS UNIQUES.

(voir pp 17-30)

* SYSUNIT : ELLE COMPREND LES PROCEDURES UTILISEES POUR RESOUDRE LES SYSTEMES D'EQUATIONS LINEAIRES.

(voir pp 31-41)

```
(*$S+,N+*)
UNIT UTILUNIT;intrinsic code 16 data 17;

INTERFACE
(*======*)
USES APPLESTUFF;

CONST DIML = 5; (* DIMension maximum en Lignes *)
      DIMC = 6; (* DIMension maximum en Colonnes *)

TYPE FRACTION = RECORD
                   NUM:INTEGER;
                   DEN:INTEGER
                END;
     (* pour enregistrer une fraction dans une seule variable *)


      SOLENTIER = RECORD
                    NUM:INTEGER;
                    DEN:INTEGER;
                    VALEUR:BOOLEAN
                  END;
     (* pour memoriser une solution sous forme de 2 entiers:1 NUMERATEUR
        et 1 DENOMINATEUR .
        VALEUR est vrai si il existe une valeur *)

      SOLREEL = RECORD
                  SOLUTION:REAL;
                  VALEUR:BOOLEAN
                END;
     (* meme principe ,mais la solution est exprimee en reel *)

     MATRICE = ARRAY [1..DIML,1..DIMC] OF FRACTION;

     MATREEL = ARRAY [1..DIML,1..DIMC] OF REAL;

     TABNBR = ARRAY [1..DIMC] OF INTEGER;

     SOLUTION = (IMPOS,INDET,UNIQUE);

     INTRO = (SYSTEME,SOLUNIQUE,SOLINDET);
     (* il est prevu de permettre l'introduction de systemes,de solutions
        uniques et de solutions indeterminees *)

     ZEROUN = 0..1;

VAR MODE,MODEDEPART :ZEROUN;
    (* MODE=0 si on travaille en FRACTION
            1 si on travaille en reels.
      MODEDEPART constitue une copie de MODE pour pouvoir representer
                 l'exercice dans son etat initial au cas ou le mode aurait
                 change pendant le calcul des solutions *)

    I,J,INDMAX,NBLIGNE,NLIGVIDE,ORDREIND,
    POSLIGNE,POSVAR,NBXELIM,NBXVAL,
    LIGNE,INDICE,ETAPE,NBETAPE,NCOLVIDE,
    NROCVIDE,POSITION,MAXADMIS : INTEGER;
    (* toutes ces variables sont utilisees dans diverses procedures de
       SYSUNIT.leur role sera plus clair au sein de ces procedures  *)

    TROUVE,IMPOSSIBLE,CVIDE,SUNFRAC :BOOLEAN;
    (* CVIDE:vrai si on a trouve une colonne vide
       SUNFRAC:vrai si la solution unique peut s'exprimer en fraction
    *)
```

```
    MAT,INTER : MATRICE;
    (* MAT:matrice de depart contenant les coefficients du systeme en
          fractions.
       INTER:matrice servant a stocker les resultats intermediaires
    *)

    MREEL,INTREEL : MATREEL;
    (* meme chose en reel *)

    XF,SFUNIQUE : ARRAY [1..DIML] OF SOLENTIER;
    (* XF : sert a stocker les valeurs des differentes variables,trouvees
          par le systeme de resolution.
       SFUNIQUE : idem mais pour les solutions introduites par l'utilisateur *)


    XR,SRUNIQUE : ARRAY [1..DIML] OF SOLREEL;
    (* idem en reel *)

    OK : ARRAY [1..DIML] OF BOOLEAN;
    (* tableau qui indique si une ligne est enregistree en fraction ou en reel*
    *)

    VIND : ARRAY[1..DIMc] OF INTEGER;
    (* Vecteur d'INDirection *)

    TYPESOL : SOLUTION;

    TYPEINTRO : INTRO;

PROCEDURE BIP;
PROCEDURE SUITE;
PROCEDURE PASSAGEREEL;
PROCEDURE COLVIDE(var vide : boolean ;
                  var indice,posindice:integer);
FUNCTION PGCD (NOMBRE:TABNBR;LONG:INTEGER):INTEGER;
PROCEDURE ATTENTION;
PROCEDURE MULTIPLICATION (NUM1,DEN1,
                          NUM2,DEN2:INTEGER;
                          VAR RESNUM,
                              RESDEN:INTEGER;
                          VAR RES:REAL;
                          VAR ERROR:ZEROUN);
PROCEDURE SOUSTRACTION (NUM1,DEN1,
                        NUM2,DEN2:INTEGER;
                        VAR RESNUM,
                            RESDEN:INTEGER;
                        VAR RES:REAL;
                        VAR ERROR:ZEROUN);
PROCEDURE EGALERMAT (NL,NC:INTEGER;
                     MF1:MATRICE;VAR MF2:MATRICE;
                     MR1:MATREEL;VAR MR2:MATREEL);
PROCEDURE CAPACITE (NBRE:REAL;VAR ADMIS:BOOLEAN);
PROCEDURE SIMPLIMAT;
PROCEDURE SIMPLISUNIQUE;
PROCEDURE DENOM (NUM,DEN : INTEGER;
                 VAR FRAC : STRING);
PROCEDURE SORTIELIGNE(NL:INTEGER;
                      MF:MATRICE;MR:MATREEL);
PROCEDURE IMPMAT(MF:MATRICE;MR:MATREEL);
PROCEDURE CHANGERORDREDESLIGNES(LIGNE,INDICE:INTEGER);
PROCEDURE ECHANGERLIGNE (L1,L2 : INTEGER );
(* ---------------------------------- *)

(* DECLARATIONS DES PROCEDURES *)
```

```
IMPLEMENTATION
(*==========*)

PROCEDURE BIP;(* emet un signal sonore *)
(***********)
BEGIN
  NOTE (41,30)
END;

PROCEDURE SUITE;
(* bloque le defilement de l'ecran jusqu'a ce qu'on tape un caractere *)
(**************)
VAR CAR:CHAR;
BEGIN
  READ(KEYBOARD,CAR)
END;

PROCEDURE PASSAGEREEL;
(*********************)
(* transforme les fractions de INTER et XF en reels et les range dans
   INTREEL et XR *)
VAR I,J:INTEGER;
BEGIN
  FOR I:=1 TO NBLIGNE
   DO FOR J:=1 TO INDMAX+1
       DO INTREEL[I,J]:=INTER[I,J].NUM/INTER[I,J].DEN;
  FOR I:=1 TO INDMAX
   DO IF XF[I].VALEUR
     THEN BEGIN XR[I].VALEUR:=TRUE;
                XR[I].SOLUTION:=XF[I].NUM/XF[I].DEN
          END
END;

procedure colvide;
(*****************)
(* verifie si une des colonnes de la zone de recherche n'est pas vide *)
var i,j:integer;
    test:real;
    trouve,fin:boolean;
begin
  j:=nbxelim+1;vide:=false;
  while (j<=(indmax-ncolvide)) and (not vide)
   do begin
        i:=posligne;fin:=false;
        while (i<=(nbligne-nligvide)) and (not fin)
         do begin
              if mode=0
               then test:=inter[i,vind[j]].num
               else test:=intreel[i,vind[j]];
              if test=0
               then begin
                      i:=i+1;
                      if i>(nbligne-nligvide)
                       then begin
                              vide:=true;
                              indice:=vind[j];
                              posindice:=j
                            end
                    end
               else
                begin j:=j+1;
                      fin:=true
                end
         end
   end
```

```
end;

(* ------------------------------ *)
FUNCTION PGCD ;
(**************)
(* elle renvoie le PGCD de LONG nombres stockes dans un tableau *)
VAR DIVIDENDE,DIVISEUR,RESTE,I,J:INTEGER;

  PROCEDURE TEST (VAR DIVISEUR,RESTE:INTEGER);
  (* sert a inverser le reste et le diviseur s'il y a lieu *)
  BEGIN
    IF DIVISEUR > RESTE
     THEN
      BEGIN
        DIVIDENDE:=DIVISEUR;
        DIVISEUR:=RESTE
      END
     ELSE
      DIVIDENDE := RESTE
  END;

(* DEBUT PRINCIPAL DE PGCD*)

BEGIN
  I:=1;
  (* la partie ci-dessous sert a eliminer les elements nuls *)
  WHILE I<=LONG
   DO IF NOMBRE[I]=0
      THEN BEGIN
             FOR J:=I+1 TO LONG
              DO NOMBRE[J-1]:=NOMBRE[J];
             LONG:=LONG-1
           END
       ELSE I:=I+1;

  IF LONG=0
   THEN PGCD:=1
   ELSE
    BEGIN
      IF LONG=1
       THEN PGCD:=NOMBRE[1]
       ELSE
        BEGIN
          DIVISEUR:=ABS(NOMBRE[1]);
          I:=2;
          WHILE (I<=LONG) AND (DIVISEUR <> 1)
          DO
           BEGIN
             RESTE:=ABS(NOMBRE[i]);
             WHILE RESTE <> 0
             DO
              BEGIN
                TEST(DIVISEUR,RESTE);
                RESTE:=DIVIDENDE MOD DIVISEUR
              END;
             I:=I+1;
           END;
          PGCD:=DIVISEUR;
        END
    END
 END;
(* ------------------------------ *)
PROCEDURE ATTENTION;
(*******************)
(* affiche un message qui signale le passage en reel *)
```

```
BEGIN
  GOTOXY(0,16);
  WRITELN('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
  WRITELN('!!! LES NOMBRES TRAITES !!!');
  WRITELN('!!! SONT TROP ELEVES :  !!!');
  WRITELN('!!! PASSAGE EN FLOTTANT !!!');
  WRITELN('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
  SUITE;
END;
(* ---------------------------- *)
PROCEDURE MULTIPLICATION ;
(**************************)
(* effectue la  MULTIPLICATION de 2 fractions.en cas de depassement de
   capacite des nombres reels un code d'erreur est positionne et le result
   est donne en reel *)
VAR DIVISEUR:INTEGER;
    TAMPREEL,BUF:REAL;
    BON:BOOLEAN;
    NBR:TABNBR;

BEGIN
  ERROR:=0;
  NBR[1]:=NUM1;
  NBR[2]:=DEN2;
  DIVISEUR:=PGCD(NBR,2);
  NUM1:=NUM1 DIV DIVISEUR;
  DEN2:=DEN2 DIV DIVISEUR;
  NBR[1]:=NUM2;
  NBR[2]:=DEN1;
  DIVISEUR:=PGCD(NBR,2);
  NUM2:=NUM2 DIV DIVISEUR;
  DEN1:=DEN1 DIV DIVISEUR;
  BUF:=NUM1;
  TAMPREEL:=BUF*NUM2;
  CAPACITE(TAMPREEL,BON);
  IF BON
   THEN
    BEGIN
      NBR[1]:=ROUND(TAMPREEL);
      BUF:=DEN1;
      TAMPREEL:=BUF*DEN2;
      CAPACITE(TAMPREEL,BON);
      IF BON
       THEN
        BEGIN
          NBR[2]:=ROUND(TAMPREEL);
          RESNUM:=NBR[1];
          RESDEN:=NBR[2];
          DIVISEUR:=PGCD(NBR,2);
          IF DIVISEUR<>1
           THEN
            BEGIN
              RESNUM:=RESNUM DIV DIVISEUR;
              RESDEN:=RESDEN DIV DIVISEUR
            END
        END
       ELSE
        BEGIN
         ERROR:=1;
         ATTENTION;
         BUF:=NBR[1];
         RES:=BUF/TAMPREEL
        END
    END
   ELSE
```

```
    BEGIN
      ERROR:=1;
      ATTENTION;
      BUF:=DEN1;
      BUF:=BUF*DEN2;
      RES:=TAMPREEL/BUF
    END
END;
(* ---------------------------------- *)
PROCEDURE SOUSTRACTION ;
(* meme remarque que pour la multiplication *)
VAR DIVISEUR : INTEGER;
    TAMPREEL,BUF1,BUF2 : REAL ;
    BON:BOOLEAN;
    NBR:TABNBR;

BEGIN
  ERROR:=0;
  NBR[1]:=DEN1;
  NBR[2]:=DEN2;
  DIVISEUR:=PGCD(NBR,2);
  BUF1:=NUM1;
  BUF1:=BUF1*DEN2;
  BUF2:=NUM2;
  BUF2:=BUF2*DEN1;
  TAMPREEL:=(BUF1-BUF2)/DIVISEUR;
  CAPACITE(TAMPREEL,BON);
  IF BON
   THEN
    BEGIN
      NBR[1]:=ROUND(TAMPREEL);
      RESNUM:=NBR[1];
      BUF1:=DEN1;
      BUF2:=BUF1*DEN2;
      TAMPREEL:=BUF2/DIVISEUR;
      CAPACITE(TAMPREEL,BON);
      IF BON
       THEN
        BEGIN
          NBR[2]:=ROUND(TAMPREEL);
          RESDEN:=NBR[2];
          DIVISEUR:=PGCD(NBR,2);
          IF DIVISEUR <> 1
           THEN
            BEGIN
              RESNUM:=RESNUM DIV DIVISEUR;
              RESDEN:=RESDEN DIV DIVISEUR
            END
        END
       ELSE
        BEGIN
          ERROR:=1;
          ATTENTION;
          BUF1:=NBR[1];
          RES:=BUF1/TAMPREEL
        END
    END
    ELSE
     BEGIN
       ERROR:=1;
       ATTENTION;
       BUF1:=DEN1;
       BUF1:=BUF1*DEN2;
       BUF1:=BUF1/DIVISEUR;
       RES:=TAMPREEL/BUF1
```

```
    END
END;
(* ------------------------------ *)
PROCEDURE EGALERMAT ;
(* permet une assignation entre matrices de fractions ou de reels *)
VAR I,J:INTEGER;

BEGIN
  FOR I:=1 TO NL
   DO
    FOR J:=1 TO NC
     DO
      IF MODE=0
       THEN
        BEGIN
          MF2[I,J].NUM:=MF1[I,J].NUM;
          MF2[I,J].DEN:=MF1[I,J].DEN
        END
       ELSE
        MR2[I,J]:=MR1[I,J]
END;
(* ------------------------------ *)
PROCEDURE CAPACITE ;
(******************)
(* verifie si un nombre reel est ou non superieur a maxint *)
BEGIN
  IF ABS(NBRE) > MAXINT
   THEN ADMIS:=FALSE
   ELSE ADMIS:=TRUE
END;
(* ------------------------------ *)

PROCEDURE SIMPLIMAT;
(* simplification d'une matrice de fractions effectuees en 2 etapes
      1:simplification des differentes fractions
      2:simplification des lignes ne contenant que des entiers *)
VAR DIVISEUR : INTEGER;
    NBR:TABNBR;
  (* --------------------------- *)
  PROCEDURE SIMPFRACTION;
  (* simplification des fractions *)
  BEGIN
    FOR I:=1 TO NBLIGNE
     DO FOR J:=1 TO INDMAX+1
         DO IF (MAT[I,J].NUM <>0) AND (MAT[I,J].DEN <> 1)
              THEN
               BEGIN
                 NBR[1]:=(MAT[I,J].NUM);
                 NBR[2]:=(MAT[I,J].DEN);
                 DIVISEUR:=PGCD(NBR,2);
                 MAT[I,J].NUM:=MAT[I,J].NUM DIV DIVISEUR;
                 MAT[I,J].DEN:=MAT[I,J].DEN DIV DIVISEUR
               END;
  END;
  (* ------------------------- *)
  PROCEDURE SIMPENTIER;
  (* simplification des lignes entieres *)
  VAR TROUVE:BOOLEAN;
      LONG,FACTREST:INTEGER;
      NBR:TABNBR;

    PROCEDURE GRANDEUR (FACTEUR:INTEGER;
                        VAR RES:INTEGER);
    (* determination du facteur restant apres elimination des puissances
       de 2,3 et 5 *)
```

```
    BEGIN
      RES:=FACTEUR;
      WHILE RES MOD 2 = 0
       DO RES:=RES DIV 2;
      WHILE RES MOD 3 = 0
       DO RES:=RES DIV 3;
      WHILE RES MOD 5 = 0
       DO RES:=RES DIV 5;
    END;
    (* ------------------------- *)
  BEGIN
    FOR I:=1 TO NBLIGNE
     DO BEGIN
          TROUVE:=FALSE;
          LONG:=0;J:=1;
          WHILE (J<=INDMAX+1) AND (NOT TROUVE)
           DO IF (MAT[I,J].DEN<>1)
              THEN TROUVE:=TRUE
              ELSE BEGIN
                     IF MAT[I,J].NUM<>0
                          THEN BEGIN
                                 LONG:=LONG+1;
                                 NBR[LONG]:=MAT[I,J].NUM
                               END;
                     J:=J+1
                   END;

          IF (NOT TROUVE)
           THEN BEGIN
                  DIVISEUR:=PGCD(NBR,LONG);
                  IF DIVISEUR<>1
                   THEN
                     BEGIN
                       FOR J:=1 TO INDMAX+1
                         DO MAT[I,J].NUM:=MAT[I,J].NUM DIV DIVISEUR;
                       GRANDEUR (DIVISEUR,FACTREST);
                       IF FACTREST>MAXADMIS
                        THEN
                         BEGIN
                           GOTOXY(0,20);
                           WRITELN ('ATTENTION:LA LIGNE',I,'EST ');
                           WRITELN ('SIMPLIFIEE PAR ',DIVISEUR);
                           SUITE;
                         END;
                     END;
                END;
        END;
  END;
  (* ------------------------- *)
BEGIN
  SIMPFRACTION;
  SIMPENTIER
END;
(* ------------------------- *)
PROCEDURE SIMPLISUNIQUE;
(* determination du "type" (reel ou fraction) de la solution unique
   introduite par l'utilisateur et en cas de solution fractionnaire
   simplification de celle-ci *)
VAR DIVISEUR:INTEGER;
    NBR:TABNBR;
BEGIN
  SUNFRAC:=TRUE;
  FOR I:=1 TO INDMAX
   DO IF NOT OK[I]
      THEN SUNFRAC:=FALSE;
```

```
  IF SUNFRAC
   THEN
    FOR I:=1 TO INDMAX
     DO IF (SFUNIQUE[I].NUM<>0)
        AND (SFUNIQUE[I].DEN<>1)
         THEN BEGIN
                NBR[1]:=SFUNIQUE[I].NUM;
                NBR[2]:=SFUNIQUE[I].DEN;
                DIVISEUR:=PGCD(NBR,2);
                SFUNIQUE[I].NUM:=SFUNIQUE[I].NUM DIV DIVISEUR;
                SFUNIQUE[I].DEN:=SFUNIQUE[I].DEN DIV DIVISEUR;
              END;
END;

 (* ---------------------------- *)
PROCEDURE DENOM;
(* sert a exprimer une fraction pour presentation a l'ecran
     par exple ne pas ecrire 2/1 mais 2
*)
VAR TAMPON1,TAMPON2,BARRE:STRING;

BEGIN
  FRAC:='';
  BARRE:='/';
  IF DEN=1
   THEN
    IF ABS(NUM)=1
     THEN
     ELSE STR(ABS(NUM),FRAC)
   ELSE
    BEGIN
      STR(ABS(NUM),TAMPON1);
      STR(DEN,TAMPON2);
      FRAC:=CONCAT(TAMPON1,BARRE,TAMPON2)
    END
END;

(* ---------------------------- *)
PROCEDURE SORTIELIGNE;
(* permet l'impression a l'ecran d'une ligne d'une matrice de coefficien
   sous forme d'equation et ce en fractions comme en reels *)
VAR SIGNE:CHAR;
    FRAC:STRING;
    J:INTEGER;
    PASSAGE:BOOLEAN;

BEGIN
  WRITELN;
  PASSAGE:=FALSE;
  IF (MODE=0) AND (OK[NL])
   THEN
   BEGIN
     FOR J:=1 TO INDMAX
     DO IF MF[NL,J].NUM<>0
         THEN
         BEGIN
           PASSAGE:=TRUE;
           IF MF[NL,J].NUM>0
           THEN SIGNE:='+'
           ELSE SIGNE:='-';
           DENOM(ABS(MF[NL,J].NUM),MF[NL,J].DEN,FRAC);
           WRITE(SIGNE,FRAC,' X',J,'');
         END;
        IF NOT PASSAGE THEN WRITE ('0 ');
        IF MF[NL,INDMAX+1].NUM=0
```

```
          THEN WRITELN('=0')
          ELSE IF MF[NL,INDMAX+1].DEN=1
                THEN WRITELN('=',MF[NL,INDMAX+1].NUM)
                ELSE WRITELN('=',MF[NL,INDMAX+1].NUM,'/',
                                 MF[NL,INDMAX+1].DEN);
    END
    ELSE
    BEGIN
      PASSAGE:=FALSE;
      FOR J:=1 TO INDMAX
      DO IF MR[NL,J]<>0
          THEN BEGIN
                 PASSAGE:=TRUE;
                 IF MR[NL,J]>0
                  THEN SIGNE:='+'
                  ELSE SIGNE:='-';
                 WRITE(SIGNE,ABS(MR[NL,J]),' X',J,'');
               END;
      IF NOT PASSAGE THEN WRITE('0 ');
      WRITELN('=',MR[NL,INDMAX+1]);
    END
END;

(* ------------------------------ *)
PROCEDURE IMPMAT;
(* impression d'une matrice de coefficients sous forme de systeme d'equa-
   tions lineaires *)
VAR I:INTEGER;
    CAR:CHAR;
BEGIN
  FOR I:=1 TO NBLIGNE
  DO SORTIELIGNE(I,MF,MR);
END;
(* ------------------------------ *)
PROCEDURE ECHANGERLIGNE;
(* echanger la position de 2 lignes.
   est utilise pour echanger une ligne "vide" avec la derniere ligne "
   "non-vide". *)
VAR LENTIERE : ARRAY [1..DIMC] OF FRACTION;
    LREELLE : ARRAY [1..DIMC] OF REAL;
    J:INTEGER;
BEGIN
  IF MODE=0
   THEN
      FOR J:=1 TO INDMAX+1
       DO
        BEGIN
          LENTIERE[J].NUM:=INTER[L2,J].NUM;
          LENTIERE[J].DEN:=INTER[L2,J].DEN;
          INTER[L2,J].NUM:=INTER[L1,J].NUM;
          INTER[L2,J].DEN:=INTER[L1,J].DEN;
          INTER[L1,J].NUM:=LENTIERE[J].NUM;
          INTER[L1,J].DEN:=LENTIERE[J].DEN
        END
    ELSE
      FOR J:=1 TO INDMAX+1
       DO
        BEGIN
          LREELLE[J]:=INTREEL[L2,J];
          INTREEL[L2,J]:=INTREEL[L1,J];
          INTREEL[L1,J]:=LREELLE[J]
        END
END;
(* ------------------------------ *)
PROCEDURE CHANGERORDREDESLIGNES;
```

```
(* sert a amener une ligne qui a permis de determiner la valeur d'une
   variable, en tete du systeme et a decaler les autres *)
VAR I,J:INTEGER;
BEGIN
  IF MODE=0
   THEN
    BEGIN
      FOR I:=LIGNE-1 DOWNTO 1
       DO
        FOR J:=1 TO INDMAX+1
         DO
          BEGIN
            INTER[I+1,J].NUM:=INTER[I,J].NUM;
            INTER[I+1,J].DEN:=INTER[I,J].DEN
          END;
      FOR J:=1 TO INDMAX
       DO
        IF J<>INDICE
         THEN INTER[1,J].NUM:=0
         ELSE
          BEGIN
            INTER[1,J].NUM:=1;
            INTER[1,J].DEN:=1
          END;
      INTER[1,INDMAX+1].NUM:=XF[INDICE].NUM;
      INTER[1,INDMAX+1].DEN:=XF[INDICE].DEN
    END
   ELSE
    BEGIN
      FOR I:=LIGNE-1 DOWNTO 1
       DO
        FOR J:=1 TO INDMAX+1
         DO
          INTREEL[I+1,J]:=INTREEL[I,J];
      FOR J:=1 TO INDMAX
       DO
        IF J<>INDICE
         THEN INTREEL[1,J]:=0
         ELSE INTREEL[1,J]:=1;
      INTREEL[1,INDMAX+1]:=XR[INDICE].SOLUTION
    END
 END;
(* ------------------------------ *)

BEGIN
END.
(* -------------------------------- *)
```

```
(*$S+,N+*)
UNIT INTROUNIT;

INTERFACE
(*======*)
USES APPLESTUFF,UTILUNIT;

TYPE MATETATOUACTION=ARRAY [0..15,1..11] OF INTEGER;
     MATMESSAGES=ARRAY [0..15] OF STRING[40];
     (* ces 2 types sont definis pour declarer la taille des matrices
        que nous utilisons
              - matrices d'etats (tt,tt2)
              - matrices d'action (ta,ta2)
              - matrices de messages d'erreur (tm,tm2)
     *)

VAR FININTRO,CONTINUER,BON,SYSTVIDE : BOOLEAN ;

    MAXIND,POSIT,K : INTEGER;

    REP:CHAR;

    INDEP : ARRAY [1..DIML] OF FRACTION;
    INDREEL : ARRAY [1..DIML] OF REAL;
    (* INDEP  et INDREEL sont les vecteurs qui contiennent les termes
       independants avant de savoir ou on les introduit dans la matrice *)

    TAMPREEL,NUMREEL,NBREEL : REAL;

    TT,TA,TT2,TA2 : MATETATOUACTION;

    TM,TM2 : MATMESSAGES;

FUNCTION EXPO(N1,N2:INTEGER):REAL;
PROCEDURE LIGAZERO(NROLIGNE:INTEGER);
PROCEDURE LIGREELLE(NROLIGNE:INTEGER);
PROCEDURE INIINTRODUCTION;
PROCEDURE ANALIGNE (NLIGNE:INTEGER;TYPEINTRO:INTRO);
PROCEDURE INTTI;
PROCEDURE INTRODUCTION(TYPEINTRO:INTRO);
PROCEDURE DETERMINERMODE;
PROCEDURE DEMANDER;
PROCEDURE CORSYS;
PROCEDURE CORSOLUNIQUE;

(* ---------------------------------- *)
IMPLEMENTATION
(*==========*)

FUNCTION EXPO;
(**********)
(* calcul d'exposant *)
VAR I : INTEGER;
   MUL : REAL;
BEGIN
 MUL:=1;
 FOR I:=N2 DOWNTO 1
  DO MUL:=MUL*N1;
 EXPO:=MUL
END;
(* ---------------------------- *)
PROCEDURE LIGAZERO;
(****************)
(* initialise une ligne donnee en mode fraction et remet les valeurs a 0 *
VAR J:INTEGER;
```

```
BEGIN
 OK[NROLIGNE]:=TRUE;
 FOR J:=1 TO DIMC
  DO
   WITH MAT[NROLIGNE,J]
    DO
     BEGIN
       NUM:=0;
       DEN:=1
     END;
 INDEP[NROLIGNE].NUM:=0;
 INDEP[NROLIGNE].DEN:=1;
END;
 (* ------------------------------ *)
PROCEDURE LIGREELLE;
(*******************)
(* transforme une ligne de fractions en reels,a cause d'un depassement
   de capacite *)
VAR J:INTEGER;
BEGIN
FOR J:=1 TO DIMC
 DO
  MREEL[NROLIGNE,J]:=MAT[NROLIGNE,J].NUM/MAT[NROLIGNE,J].DEN
END;
 (* ------------------------------ *)
PROCEDURE INIINTRODUCTION;
(*************************)
  (* realise l'initialisation des matrices
     INIT1 a INIT7 *)

  PROCEDURE INIT1;
  VAR I,J:INTEGER;
  BEGIN
    FOR I:=0 TO 15
     DO
      BEGIN
        FOR J:=1 TO 9
         DO
          BEGIN
            TA[I,J]:=99;TA2[I,J]:=99;
          END;
        TT[I,10]:=I;TT[I,11]:=0;
        TT2[I,10]:=I;TT2[I,11]:=0;
        TA[I,10]:=0;TA[I,11]:=98;
        TA2[I,10]:=0;TA2[I,11]:=98;
      END
  END;
  (* ------------------------------ *)
  PROCEDURE INIT2;
  BEGIN
    TT[0,1]:=1;TT[0,3]:=5;TT[0,4]:=2;TT[0,5]:=2;TT[0,6]:=3;TT[0,8]:=16;
    TT[1,1]:=1;TT[1,2]:=4;TT[1,3]:=5;TT[1,6]:=3;
    TT[2,1]:=1;TT[2,3]:=5;TT[2,6]:=3;
    TT[3,1]:=6;
    TT[4,1]:=7;
    TT[5,1]:=8;
    TT[6,4]:=2;TT[6,5]:=2;TT[6,7]:=9;
    TT[7,1]:=7;TT[7,6]:=3;
    TT[8,1]:=8;TT[8,6]:=3;
    TT[9,1]:=10;TT[9,3]:=13;TT[9,4]:=11;TT[9,5]:=11;
    TT[10,1]:=10;TT[10,2]:=12;TT[10,3]:=13;TT[10,8]:=16;
    TT[11,1]:=10;TT[11,3]:=13;
    TT[12,1]:=14;
    TT[13,1]:=15;
    TT[14,1]:=14;TT[14,8]:=16;
```

```
  TT[15,1]:=15;TT[15,8]:=16
END;
(* ---------------------------------- *)
PROCEDURE INIT3;
BEGIN
  TA[0,1]:=1;TA[0,3]:=20;TA[0,4]:=2;TA[0,5]:=3;TA[0,6]:=4;TA[0,8]:=18;
  TA[1,1]:=5;TA[1,2]:=6;TA[1,3]:=7;TA[1,6]:=8;
  TA[2,1]:=1;TA[2,3]:=20;TA[2,6]:=4;
  TA[3,1]:=9;
  TA[4,1]:=21;
  TA[5,1]:=10;
  TA[6,4]:=2;TA[6,5]:=3;TA[6,7]:=19;
  TA[7,1]:=14;TA[7,6]:=11;
  TA[8,1]:=12;TA[8,6]:=13;
  TA[9,1]:=1;TA[9,3]:=20;TA[9,4]:=2;TA[9,5]:=3;
  TA[10,1]:=5;TA[10,2]:=6;TA[10,3]:=7;TA[10,8]:=15;
  TA[11,1]:=1;TA[11,3]:=20;
  TA[12,1]:=21;
  TA[13,1]:=10;
  TA[14,1]:=14;TA[14,8]:=16;
  TA[15,1]:=12;TA[15,8]:=17
END;
(* ---------------------------------- *)
PROCEDURE INIT4;
VAR I:INTEGER;
BEGIN
  FOR I:=0 TO 15 DO TM[I]:='';
  TM[0]:=' UN CHIFFRE  .  ,  +  -  X  RET ';
  TM[1]:=' UN CHIFFRE  /  .  ,  X ';
  TM[2]:=' UN CHIFFRE  .  ,  X ';
  TM[3]:=' UNIQUEMENT UN CHIFFRE <= 5 ';
  TM[4]:=' UNIQUEMENT UN CHIFFRE ';
  TM[5]:=' UNIQUEMENT UN CHIFFRE ';
  TM[6]:=' +  -  = ';
  TM[7]:=' UN CHIFFRE  OU  X ';
  TM[8]:=' UN CHIFFRE  OU  X ';
  TM[9]:=' UN CHIFFRE  .  ,  +  - ';
  TM[10]:=' UN CHIFFRE  /  .  ,  RET ';
  TM[11]:=' UN CHIFFRE  .  , ';
  TM[12]:=' UNIQUEMENT UN CHIFFRE ';
  TM[13]:=' UNIQUEMENT UN CHIFFRE ';
  TM[14]:=' UN CHIFFRE  OU  RET ';
  TM[15]:=' UN CHIFFRE  OU  RET ';
END;
(* ---------------------------------- *)
PROCEDURE INIT5;
BEGIN
  TT2[0,1]:=2;TT2[0,3]:=3;TT2[0,4]:=1;TT2[0,5]:=1;
  TT2[1,1]:=2;TT2[1,3]:=3;
  TT2[2,1]:=2;TT2[2,2]:=4;TT2[2,3]:=3;TT2[2,8]:=16;
  TT2[3,1]:=5;
  TT2[4,1]:=6;
  TT2[5,1]:=5;TT2[5,8]:=16;
  TT2[6,1]:=6;TT2[6,8]:=16;
END;
(* ---------------------------------- *)
PROCEDURE INIT6;
BEGIN
  TA2[0,1]:=1;TA2[0,3]:=20;TA2[0,4]:=2;TA2[0,5]:=3;
  TA2[1,1]:=1;TA2[1,3]:=20;
  TA2[2,1]:=5;TA2[2,2]:=6;TA2[2,3]:=7;TA2[2,8]:=31;
  TA2[3,1]:=10;
  TA2[4,1]:=21;
  TA2[5,1]:=12;TA2[5,8]:=32;
  TA2[6,1]:=14;TA2[6,8]:=33;
```

```
  END;
  (* ------------------------------ *)
  PROCEDURE INIT7;
  VAR I:INTEGER;
  BEGIN
    FOR I:=0 TO 15 DO TM2[I]:='';
    TM2[0]:=' UN CHIFFRE  .  ,  +  -  ';
    TM2[1]:=' UN CHIFFRE  .  ,';
    TM2[2]:=' UN CHIFFRE  /  .  ,  RET ';
    TM2[3]:=' UNIQUEMENT UN CHIFFRE ';
    TM2[4]:=' UNIQUEMENT UN CHIFFRE ';
    TM2[5]:=' UN CHIFFRE  OU   RET ';
    TM2[6]:=' UN CHIFFRE  OU   RET ';
  END;
 (* ------------------------ *)
BEGIN
  INIT1;INIT2;INIT3;INIT4;
  INIT5;INIT6;INIT7;
END;
(* ------------------------------ *)
PROCEDURE ANALIGNE;
(* analyse une ligne introduite par l'utilisateur.
   en fonction des conventions definies,elle refuse ou accepte et memorise
   chaque caractere. *)
VAR ETAT,SIGNE,NOMBRE,NUM,DEN,TYPECRT,
    PARTENT,CPTDEC,ACTION : INTEGER ;
    BUF:REAL;
    ERREUR : BOOLEAN;
    CAR : CHAR;
(* ------------------------------ *)
PROCEDURE NEXTCAR (VAR CARACTERE:CHAR;
                   POSITION : INTEGER;
                   VAR TYP : INTEGER);
(* lit un caractere et renvoie son type sous forme de code chiffre
   (efface egalement le message d'erreur precedent s'il y a lieu) *)
BEGIN
  GOTOXY(POSITION,2+NLIGNE);
  READ(KEYBOARD,CARACTERE);
  IF ERREUR
   THEN
    BEGIN
      GOTOXY(0,20);
      WRITELN ('                                                  ');
      WRITELN ('                                                  ');
      WRITE ('                                                ');
    END;
  TYP:=9;(* SI LE TYPE N'EST PAS MODIFIE -> CAR INVALIDE *)
  IF EOLN(KEYBOARD)
   THEN TYP:=8 (* <RET> *)
   ELSE CASE ORD(CARACTERE) OF
        48,49,50,51,52,53,54,55,56,57 : TYP:=1;
        47 : TYP:=2;(* / *)
        44,46 : TYP:=3;(* . , *)
        43 : TYP:=4;(* + *)
        45 : TYP:=5;(*-*)
        88,120 : TYP:=6;(* X *)
        61 : TYP:=7;
        32 : TYP:=10;(*   *)
        8 : TYP:=11 (* FLECHE < *)
        END
END;
(* ------------------------------ *)
(*
 * toutes les actions sont decrites dans le memoire-meme
 *)
```

```
PROCEDURE ACT5;
BEGIN
  IF OK[NLIGNE]
   THEN
    BEGIN
      BUF:=NOMBRE;
      TAMPREEL:=(BUF*10)+ORD(CAR)-48;
      CAPACITE (TAMPREEL,OK[NLIGNE]);
      IF OK[NLIGNE]
       THEN NOMBRE:=ROUND(TAMPREEL)
       ELSE
        BEGIN
          NBREEL:=TAMPREEL;
          CASE TYPEINTRO OF
           SYSTEME:LIGREELLE(NLIGNE);
           SOLUNIQUE:OK[NLIGNE]:=FALSE;
          END;
        END;
    END
   ELSE
    NBREEL:=(NBREEL*10)+ORD(CAR)-48
END;
(* ------------------------------ *)
  PROCEDURE ACT9;
  BEGIN
    INDICE:=ORD(CAR)-48;
    IF (INDICE < 1) OR (INDICE > 5)
     THEN
      BEGIN
        ERREUR:=TRUE;
        BIP;
        GOTOXY(0,20);
        WRITELN ('L''INDICE DOIT APPARTENIR A (1..5)')
      END
     ELSE
      BEGIN
        IF INDICE>INDMAX
         THEN INDMAX:=INDICE;
        IF OK[NLIGNE]
         THEN
          BEGIN
            MAT[NLIGNE,INDICE].NUM:=NUM;
            MAT[NLIGNE,INDICE].DEN:=DEN;
          END
         ELSE
          MREEL[NLIGNE,INDICE]:=NBREEL
      END
  END;
(* ------------------------------ *)
 PROCEDURE ACT12;
 BEGIN
      CPTDEC:=CPTDEC+1;
      IF OK[NLIGNE]
       THEN
        BEGIN
          BUF:=NOMBRE;
          TAMPREEL:=(BUF*10)+ORD(CAR)-48;
          CAPACITE(TAMPREEL,OK[NLIGNE]);
          IF OK[NLIGNE]
          THEN
              NOMBRE:=ROUND(TAMPREEL)
           ELSE
            BEGIN
              NBREEL:=PARTENT+(TAMPREEL/EXPO(10,CPTDEC));
```

```
                CASE TYPEINTRO OF
                 SYSTEME:LIGREELLE(NLIGNE);
                 SOLUNIQUE:OK[NLIGNE]:=FALSE;
                END;
              END;
          END
         ELSE
          NBREEL:=NBREEL+((ORD(CAR)-48)/EXPO(10,CPTDEC))
     END;
    (* ------------------------------ *)
    PROCEDURE ACT13;
    BEGIN
         IF OK[NLIGNE]
          THEN
           BEGIN
             TAMPREEL:=EXPO(10,CPTDEC);
             CAPACITE(TAMPREEL,OK[NLIGNE]);
             IF OK[NLIGNE]
              THEN
               BEGIN
                 DEN:=ROUND(TAMPREEL);
                 TAMPREEL:=(PARTENT*TAMPREEL)+(SIGNE*NOMBRE);
                 CAPACITE (TAMPREEL,OK[NLIGNE]);
                 IF OK[NLIGNE]
                  THEN
                     NUM:=ROUND(TAMPREEL)
                  ELSE
                     BEGIN
                       LIGREELLE(NLIGNE);
                       NBREEL:=TAMPREEL/DEN
                     END
               END
              ELSE
               BEGIN
                 LIGREELLE(NLIGNE);
                 NBREEL:=PARTENT+((SIGNE*NOMBRE)/TAMPREEL)
               END
           END
       END;
(* ------------------------------ *)
PROCEDURE ACT14;
BEGIN
  IF OK[NLIGNE]
   THEN
    BEGIN
      BUF:=NOMBRE;
      TAMPREEL:=(BUF*10)+ORD(CAR)-48;
      CAPACITE (TAMPREEL,OK[NLIGNE]);
      IF OK[NLIGNE]
       THEN NOMBRE:=ROUND(TAMPREEL)
       ELSE
        BEGIN
          NUMREEL:=NUM;
          NBREEL:=TAMPREEL;
          CASE TYPEINTRO OF
           SYSTEME:LIGREELLE(NLIGNE);
           SOLUNIQUE:OK[NLIGNE]:=FALSE;
          END;
        END;
    END
   ELSE
    NBREEL:=(NBREEL*10)+ORD(CAR)-48
 END;
(* ------------------------------ *)
PROCEDURE ACT17;
```

```
BEGIN
     MAXIND:=INDMAX;
     IF OK[NLIGNE]
      THEN
       BEGIN
         TAMPREEL:=EXPO(10,CPTDEC);
         CAPACITE(TAMPREEL,OK[NLIGNE]);
         IF OK[NLIGNE]
          THEN
           BEGIN
             INDEP[NLIGNE].DEN:=ROUND(TAMPREEL);
             TAMPREEL:=(PARTENT*TAMPREEL)+(SIGNE*NOMBRE);
             CAPACITE(TAMPREEL,OK[NLIGNE]);
             IF OK[NLIGNE]
              THEN INDEP[NLIGNE].NUM:=ROUND(TAMPREEL)
              ELSE
               BEGIN
                 LIGREELLE(NLIGNE);
                 INDREEL[NLIGNE]:=TAMPREEL/EXPO(10,CPTDEC)
               END
           END
          ELSE
           BEGIN
             LIGREELLE(NLIGNE);
             INDREEL[NLIGNE]:=PARTENT+((SIGNE*NOMBRE)/TAMPREEL)
           END
       END
      ELSE
       INDREEL[NLIGNE]:=NBREEL
 END;
(* ------------------------------ *)
PROCEDURE ACT31;
BEGIN
  IF OK[NLIGNE]
   THEN BEGIN
          SFUNIQUE[INDICE].NUM:=SIGNE*NOMBRE;
          SFUNIQUE[INDICE].DEN:=1;
        END
   ELSE SRUNIQUE[INDICE].SOLUTION:=SIGNE*NBREEL;
END;
(* ------------------------------ *)
PROCEDURE ACT32;
BEGIN
  IF OK[NLIGNE]
   THEN
    BEGIN
      TAMPREEL:=EXPO(10,CPTDEC);
      CAPACITE(TAMPREEL,OK[NLIGNE]);
      IF OK[NLIGNE]
       THEN
        BEGIN
          DEN:=ROUND(TAMPREEL);
          TAMPREEL:=(PARTENT*TAMPREEL)+(SIGNE*NOMBRE);
          CAPACITE (TAMPREEL,OK[NLIGNE]);
          IF OK[NLIGNE]
           THEN
            BEGIN
              SFUNIQUE[INDICE].NUM:=ROUND(TAMPREEL);
              SFUNIQUE[INDICE].DEN:=DEN
            END
           ELSE SRUNIQUE[INDICE].SOLUTION:=TAMPREEL/DEN
        END
       ELSE
          SRUNIQUE[INDICE].SOLUTION:=PARTENT+((SIGNE*NOMBRE)/TAMPREEL)
    END
```

```
END;
(* ------------------------------ *)
PROCEDURE ACT33;
BEGIN
  IF OK[NLIGNE]
   THEN BEGIN
          SFUNIQUE[INDICE].NUM:=NUM;
          SFUNIQUE[INDICE].DEN:=NOMBRE
        END
   ELSE SRUNIQUE[INDICE].SOLUTION:=NUMREEL/NBREEL;
END;
(* ------------------------------ *)
PROCEDURE ACT98;(* EFFACEMENT *)
BEGIN
  IF TYPEINTRO=SYSTEME
   THEN BEGIN
          LIGAZERO(NLIGNE);
          POSIT:=0;
          INDMAX:=MAXIND
        END
   ELSE POSIT:=4;
  SIGNE:=1;
  GOTOXY(POSIT,2+NLIGNE);
  WRITE('                                                    ');
END;
(* ------------------------------ *)
PROCEDURE ACT99;
BEGIN
  ERREUR:=TRUE;
  BIP;
  GOTOXY(0,20);
  WRITELN ('CE CARACTERE EST INADEQUAT:TAPEZ');
  WRITELN ('UN DES CARACTERES SUIVANTS : ');
  CASE TYPEINTRO OF
   SYSTEME:WRITE(TM[ETAT]);
   SOLUNIQUE:WRITE(TM2[ETAT]);
  END;
END;
(* ------------------------------ *)

BEGIN

  ETAT:=0;
  SIGNE:=1;
  WHILE ETAT <> 16
   DO
    BEGIN
      NEXTCAR(CAR,POSIT,TYPECRT);
      ERREUR:=FALSE;
      CASE TYPEINTRO OF
       SYSTEME:ACTION:=TA[ETAT,TYPECRT];
       SOLUNIQUE:ACTION:=TA2[ETAT,TYPECRT];
      END;
      CASE ACTION OF

       0 :; (* RIEN *)

       1 :BEGIN
            IF OK[NLIGNE]
             THEN
              NOMBRE:=ORD(CAR)-48
             ELSE
              NBREEL:=ORD(CAR)-48
          END;
```

```
2 :SIGNE:=1;

3 :SIGNE:=-1;

4 :BEGIN
     IF OK[NLIGNE]
      THEN
       BEGIN
         NUM:=SIGNE;
         DEN:=1
       END
      ELSE
       NBREEL:=SIGNE
   END;


5 :ACT5;

6 :BEGIN
     IF OK[NLIGNE]
      THEN
       NUM:=SIGNE*NOMBRE
      ELSE
       NUMREEL:=SIGNE*NBREEL
   END;

7 :BEGIN
     IF OK[NLIGNE]
      THEN
       PARTENT:=SIGNE*NOMBRE
   END;

8 :BEGIN
     IF OK[NLIGNE]
      THEN
       BEGIN
         NUM:=SIGNE*NOMBRE;
         DEN:=1
       END
      ELSE
       NBREEL:=SIGNE*NBREEL
   END;

9 :ACT9;

10:BEGIN
      CPTDEC:=1;
      IF OK[NLIGNE]
       THEN
        NOMBRE:=ORD(CAR)-48
       ELSE
        NBREEL:=NBREEL+((ORD(CAR)-48)/10)
    END;

11:BEGIN
     IF OK[NLIGNE]
      THEN DEN:=NOMBRE
      ELSE
       NBREEL:=NUMREEL/NBREEL
   END;

12:ACT12;

13:ACT13;
```

```
14:ACT14;

15:BEGIN
     MAXIND:=INDMAX;
     IF OK[NLIGNE]
      THEN
       BEGIN
         INDEP[NLIGNE].NUM:=SIGNE*NOMBRE;
         INDEP[NLIGNE].DEN:=1
       END
      ELSE
         INDREEL[NLIGNE]:=SIGNE*NBREEL
   END;

16:BEGIN
     MAXIND:=INDMAX;
     IF OK[NLIGNE]
      THEN
       BEGIN
         INDEP[NLIGNE].NUM:=NUM;
         INDEP[NLIGNE].DEN:=NOMBRE
       END
      ELSE
       INDREEL[NLIGNE]:=NUMREEL/NBREEL
   END;

 17:ACT17;

 18:FININTRO:=TRUE;

 19:SIGNE:=1;

 20:BEGIN
      IF OK[NLIGNE]
       THEN
        PARTENT:=0
       ELSE
        NBREEL:=0
    END;

 21:BEGIN
      IF ORD(CAR)-48=0
       THEN
        BEGIN
          ERREUR:=TRUE;
          GOTOXY(0,20);
          WRITELN('ATTENTION!! LE DENOMINATEUR');
          WRITE('DOIT ETRE DIFFERENT DE 0')
        END
       ELSE
        IF OK[NLIGNE]
         THEN NOMBRE:=ORD(CAR)-48
         ELSE NBREEL:=ORD(CAR)-48
    END;

 31:ACT31;

 32:ACT32;

 33:ACT33;

 98:ACT98;

 99:ACT99;
```

```
        END;

        IF NOT ERREUR
         THEN
          BEGIN
            GOTOXY(POSIT,2+NLIGNE);
            WRITE(CAR);
            POSIT:=POSIT+1;
            CASE TYPEINTRO OF
             SYSTEME:ETAT:=TT[ETAT,TYPECRT];
             SOLUNIQUE:ETAT:=TT2[ETAT,TYPECRT];
            END;
          END
    END
END;

(* ------------------------------- *)
PROCEDURE INTTI;
(***************)
(* introduit les termes independants (contenus dans INDEP ou INDREEL)
   dans la matrice MAT ou MREEL    *)
   VAR I:INTEGER;
BEGIN
   FOR I:=1 TO NBLIGNE
    DO
     IF OK[I]
      THEN
       WITH MAT[I,INDMAX+1]
        DO
         BEGIN
           NUM:=INDEP[I].NUM;
           DEN:=INDEP[I].DEN
         END
      ELSE
       MREEL[I,INDMAX+1]:=INDREEL[I];
 END;
(* ------------------------------ *)
PROCEDURE INTRODUCTION;
(**********************)
(* gere l'analyse complete de l'introduction d'un systeme ou d'une soluit
   unique  *)
BEGIN
  CASE TYPEINTRO OF
   SYSTEME :
    BEGIN
      FININTRO:=FALSE;
      SYSTVIDE:=FALSE;
      WRITELN('VEUILLEZ INTRODUIRE VOTRE SYSTEME.');
      NBLIGNE:=0;
      MODE:=0;
      MAXADMIS:=7;
      INDMAX:=0;MAXIND:=0;
      WHILE NOT FININTRO
       DO
        BEGIN
          NBLIGNE:=NBLIGNE+1;
          IF NBLIGNE>5
           THEN
             BEGIN
               GOTOXY(0,20);
               WRITELN('COMME LE NOMBRE MAXIMUM D''EQUATIONS = 5,');
               WRITELN('L''INTRODUCTION DU SYSTEME EST TERMINEE.');
               FININTRO:=TRUE;
               SUITE
             END
```

```
            ELSE
             BEGIN
               LIGAZERO(NBLIGNE);
               POSIT:=1;
               ANALIGNE(NBLIGNE,SYSTEME)
             END;
         END;
       NBLIGNE:=NBLIGNE-1;
       IF NBLIGNE=0 THEN SYSTVIDE:=TRUE
                    ELSE INTTI;
       END;

    SOLUNIQUE:
     BEGIN
       WRITELN('VEUILLEZ INTRODUIRE LA SOLUTION UNIQUE.');
       INDICE:=0;
       FOR INDICE:=1 TO INDMAX
        DO
         BEGIN
           GOTOXY(1,2+INDICE);
           WRITE('X',INDICE,'=');
           OK[INDICE]:=TRUE;
           POSIT:=4;
           ANALIGNE(INDICE,SOLUNIQUE);
         END;
     END;
  END;
END;
(* ---------------------------- *)
PROCEDURE DETERMINERMODE;
(************************)
(* determine le mode (fraction ou reel) dans lequel les coefficients ont
   ete enregistres .
   s'il s'agit du mode fraction , transforme les lignes qui avaient pu
   etre enregistrees en fractions , en reels *)
VAR I:INTEGER;

BEGIN
   I:=1;
   WHILE (I<= NBLIGNE) AND (MODE=0)
   DO
    IF  OK[I]
     THEN I:=I+1
     ELSE MODE:=1;

   IF MODE=1
    THEN
     FOR I:=1 TO NBLIGNE
      DO
       IF OK[I]
        THEN LIGREELLE(I)
END;
(* ---------------------------- *)

PROCEDURE DEMANDER;
(******************)
(* demande a l'utilisateur si il ne veut rien modifier *)
BEGIN
  PAGE(OUTPUT);
  IMPMAT(MAT,MREEL);
  GOTOXY(0,20);
  WRITELN ('EST-CE BIEN CE SYSTEME QUE VOUS VOULIEZ ');
  WRITELN ('INTRODUIRE ? (N->NON)');
  READ (KEYBOARD,REP);
  IF (ord(REP)<>110) AND (ord(REP)<>78)
```

```
   THEN CONTINUER:=FALSE;
END;
(* ------------------------------ *)
PROCEDURE CORSYS;
(****************)
(* permet la correction d'un systeme deja introduit *)
VAR BON:BOOLEAN;
    TAMPON:INTEGER;

BEGIN
  BON:=FALSE;
  WHILE NOT BON
   DO
    BEGIN
      BIP;
      WRITELN;
      WRITELN('QUELLE LIGNE VOULEZ-VOUS MODIFIER ?');
      READ (KEYBOARD,rep);
      I:=ORD(REP)-48;
      IF (I<=0) OR (I>NBLIGNE)
       THEN
        WRITELN ('LE NRO DE LIGNE DOIT ETRE >0 ET <=',NBLIGNE)
       ELSE
        BON:=TRUE;
    END;
  TAMPON:=INDMAX;
  PAGE(OUTPUT);
  WRITE('VEUILLEZ INTRODUIRE LA LIGNE ',I);
  LIGAZERO(I);
  POSIT:=1;
  ANALIGNE(I,SYSTEME);
  IF TAMPON<INDMAX
   THEN
    BEGIN
      FOR J:=1 TO NBLIGNE
       DO
        IF J<>I
         THEN
          FOR K:=TAMPON+1 TO INDMAX
           DO
            IF OK[J]
             THEN MAT[J,K].NUM:=0
             ELSE MREEL[J,K]:=0;
      INTTI;
    END
   ELSE
    IF OK[I]
     THEN
      BEGIN
        MAT[I,INDMAX+1].NUM:=INDEP[I].NUM;
        MAT[I,INDMAX+1].DEN:=INDEP[I].DEN
      END
     ELSE MREEL[I,INDMAX+1]:=INDREEL[I];
END;

(* ------------------------------ *)
PROCEDURE CORSOLUNIQUE;
(**********************)
(* permet la correction de la solution unique introduite *)
VAR BON:BOOLEAN;
BEGIN
  PAGE(OUTPUT);
  FOR I:=1 TO INDMAX
  DO IF OK[I]
      THEN BEGIN
```

```
            WRITE('X',I,' = ',SFUNIQUE[I].NUM);
            IF SFUNIQUE[I].DEN=1
             THEN WRITELN
             ELSE WRITELN('/',SFUNIQUE[I].DEN);
          END
     ELSE WRITELN('X',I,' = ',SRUNIQUE[I].SOLUTION);
 GOTOXY(0,10);
 WRITELN('EST-CE BIEN LA SOLUTION QUE VOUS');
 WRITELN('VOULIEZ INTRODUIRE? (N --> NON) ');
 READ(KEYBOARD,REP);
 IF (ORD(REP)=110) OR (ORD(REP)=78)
  THEN BEGIN
         BON:=FALSE;
         WHILE NOT BON
         DO BEGIN
              WRITELN('DONNEZ LE NUMERO DE LA VARIABLE');
              WRITELN('A MODIFIER.');
              READ(KEYBOARD,REP);
              I:=ORD(REP)-48;
              IF (I<1) OR (I>INDMAX)
               THEN
                BEGIN
                  BIP;
                  WRITELN('L''INDICE DE LA VARIABLE DOIT');
                  WRITELN('ETRE COMPRIS ENTRE 1 ET ',INDMAX);
                END
               ELSE BON:=TRUE
            END;
         PAGE(OUTPUT);
         GOTOXY(1,2+I);
         WRITE('X',I,'=');
         OK[I]:=TRUE;
         POSIT:=4;
         INDICE:=I;
         ANALIGNE(INDICE,SOLUNIQUE);
       END
  ELSE CONTINUER:=FALSE
END;
(* ------------------------------- *)
 BEGIN
 END.
 (* ------------------------------- *)
```

```
(*$S++,N+*)
UNIT SYSUNIT;intrinsic code 25;
INTERFACE
USES APPLESTUFF,UTILUNIT;
(*  REM : ESSAI SANS GRAPHIQUE
     TURTLEGRAPHICS,DESSIN22,
     DECLAR33,CHOIXPLAN,INTER33,DESSIN33;*)

PROCEDURE RESOLUTION(PRESENTATION,INDGRAPH,INDCHEMIN:ZEROUN);
(* ---------------------------------- *)
IMPLEMENTATION
(*==========*)
PROCEDURE RESOLUTION;
(* resoud le systeme d'equations lineaires dont les coefficients ont ete
   enregistres danS MATRICE ou MATREEL ,selon le mode.
   ses 3 parametres servent uniquement a savoir comment presenter les r
   resultats *)

VAR RIENAELIMINER : BOOLEAN;

(* ------------------------------ *)
(* la procedure ci-dessous est en commentaires puisque nous n'avos pas
   la place pour executer.il en sera de meme pour tous les appels a des
   graphiques. *)

(*PROCEDURE GRAPHIQUE(MF:MATRICE;MR:MATREEL);
BEGIN
  IF (NBLIGNE=2) AND (INDMAX=2)
   THEN SYST22(MF,MR)
   ELSE IF (NBLIGNE=3) AND (INDMAX=3)
         THEN SYST33(MF,MR)
         ELSE BEGIN
                PAGE(OUTPUT);TEXTMODE;GOTOXY(0,10);
                WRITELN('DESOLE,IL N''Y A DE REPRESENTATION GRA-');
                WRITELN('PHIQUE QUE POUR LES SYSTEMES 2*2 ET 3*3.');
                WRITELN('****************************************');
              END
END;*)
(* ------------------------------ *)
PROCEDURE SOLIMPOS;
(* affiche un message signalant que ce systeme est impossible *)
BEGIN
  TYPESOL:=IMPOS;
  IF PRESENTATION=1
  THEN BEGIN
         BIP;PAGE(OUTPUT);GOTOXY (0,10);
         WRITELN('**************************************');
         WRITELN('* CE SYSTEME EST IMPOSSIBLE :        *');
         WRITELN('* EN EFFET ,AU COURS DE NOS CALCULS  *');
         WRITELN('* UNE EQUATION A ETE TRANSFORMEE EN  *');
         WRITELN('* EQUATION IMPOSSIBLE.(0=D)          *');
         WRITELN('* -->IL N''Y A DONC PAS DE SOLUTION.  *');
         WRITELN('**************************************');
         SUITE;
         (*IF (INDCHEMIN=0) AND (INDGRAPH=1)
          THEN GRAPHIQUE(MAT,MREEL);*)
       END
END;
(* ------------------------------ *)
PROCEDURE YATILXVALEUR (VAR TROUVE,IMPOSSIBLE:BOOLEAN;
                        VAR LIGNE,INDICE,posindice:INTEGER);
(* cherche si il existe dans la zone de travail une equation qui permet
   de determiner la valeur d'une variable *)
VAR I,J,NBCOEF:INTEGER;
    TEST : REAL;
```

```
BEGIN
  I:=POSLIGNE;NBCOEF:=0;TROUVE:=FALSE;
  LIGNE:=0;INDICE:=0;
  WHILE (I<=(NBLIGNE-NLIGVIDE))
    AND (NOT TROUVE) AND (NOT IMPOSSIBLE)
    DO BEGIN
       FOR J:=NBXELIM+1 TO INDMAX
        DO BEGIN
           IF MODE=0 THEN TEST:=INTER[I,VIND[J]].NUM
                     ELSE TEST:=INTREEL[I,VIND[J]];
           IF TEST <> 0
            THEN BEGIN
                   NBCOEF:=NBCOEF+1;
                   INDICE:=VIND[J];
                   posindice:=j
                 END
       END;
       IF NBCOEF=0
        THEN BEGIN
               INDICE:=0;
               IF MODE=0 THEN TEST:=INTER[I,INDMAX+1].NUM
                         ELSE TEST:=INTREEL[I,INDMAX+1];
               IF TEST = 0
                THEN BEGIN
                       ECHANGERLIGNE(I,(NBLIGNE-NLIGVIDE));
                       NLIGVIDE:=NLIGVIDE+1;
                       NBETAPE:=NBETAPE-1;
                       I:=I+1
                     END
                ELSE BEGIN
                       IMPOSSIBLE:=TRUE;
                       SOLIMPOS;
                     END
          END
        ELSE
         IF NBCOEF=1
          THEN BEGIN
                 TROUVE:=TRUE;
                 LIGNE:=I
               END
          ELSE BEGIN
                 INDICE:=0;
                 NBCOEF:=0;
                 I:=I+1
               END
       END
END;
(* ------------------------------- *)
PROCEDURE DEBUTIND(NROCOL,POSITION:INTEGER);
(* place un element dans la zone du debut du vecteur d'indirection *)
VAR J:INTEGER;
BEGIN
  IF POSITION=NBXELIM
   THEN  (* RIEN --> VECTEUR EN BON ORDRE *)
   ELSE BEGIN
          FOR J:=POSITION-1 DOWNTO NBXELIM
           DO VIND[J+1]:=VIND[J];
          VIND[NBXELIM]:=NROCOL
        END
END;
(* ------------------------------- *)
PROCEDURE FININD(NROCVIDE,POSITION:INTEGER);
(* inverse de DEBUTIND *)
VAR J:INTEGER;
BEGIN
```

```
  IF POSITION=(INDMAX-NCOLVIDE)
   THEN
   ELSE BEGIN
          FOR I:=POSITION TO INDMAX-NCOLVIDE-1
           DO VIND[I]:=VIND[I+1];
          VIND[INDMAX-NCOLVIDE]:=NROCVIDE
        END
END;
(* ------------------------------ *)
PROCEDURE DONNERVALEUR(LIGNE,INDICE:INTEGER);
(* donne une valeur a une variable ,a partir d'une equation selectionnee
   par YATIXVALEUR *)
VAR TAMPON1,TAMPON2:INTEGER;
    RES:REAL;
    CODEERREUR:ZEROUN;
BEGIN
  IF MODE=0
   THEN BEGIN
          XF[INDICE].VALEUR:=TRUE;
          IF INTER[LIGNE,INDICE].NUM<0
           THEN BEGIN
                  TAMPON1:=-INTER[LIGNE,INDICE].NUM;
                  TAMPON2:=-INTER[LIGNE,INDICE].DEN
                END
           ELSE BEGIN
                  TAMPON1:=INTER[LIGNE,INDICE].NUM;
                  TAMPON2:=INTER[LIGNE,INDICE].DEN
                END;
          MULTIPLICATION (INTER[LIGNE,INDMAX+1].NUM,INTER[LIGNE,INDMAX+1].
                          TAMPON2,TAMPON1,XF[INDICE].NUM,XF[INDICE].DEN,RE
          IF CODEERREUR=1
           THEN BEGIN
                  MODE:=1;PASSAGEREEL;
                  XR[INDICE].VALEUR:=TRUE;
                  XR[INDICE].SOLUTION:=RES
                END
        END
   ELSE BEGIN
          XR[INDICE].VALEUR:=TRUE;
          XR[INDICE].SOLUTION:=(INTREEL[LIGNE,INDMAX+1]/INTREEL[LIGNE,INDI
        END
 END;
(* ------------------------------ *)
PROCEDURE INJECTER (INDICE,NBX:INTEGER);
(* lorsqu'on a trouve la valeur d'une variable on elimine celle-ci en
   injectant sa valeur dans les autres equations *)
VAR I,NUM,DEN : INTEGER;
    RES : REAL;
    CODEERREUR:ZEROUN;
BEGIN
  FOR I:=NBX+1 TO NBLIGNE-NLIGVIDE
   DO IF MODE=0
      THEN
      BEGIN
        IF INTER[I,INDICE].NUM<>0
         THEN
          BEGIN
            MULTIPLICATION(INTER[I,INDICE].NUM,INTER[I,INDICE].DEN,
                           XF[INDICE].NUM,XF[INDICE].DEN,
                           NUM,DEN,RES,CODEERREUR);
            IF CODEERREUR=0
             THEN
              SOUSTRACTION(INTER[I,INDMAX+1].NUM,INTER[I,INDMAX+1].DEN,
                           NUM,DEN,INTER[I,INDMAX+1].NUM,INTER[I,INDMAX+1]
                           RES,CODEERREUR)
```

```
                ELSE
                 RES:=(INTER[I,INDMAX+1].NUM/INTER[I,INDMAX+1].DEN)-RES;
               INTER[I,INDICE].NUM:=0;
               IF CODEERREUR=1
                THEN BEGIN
                       MODE:=1;PASSAGEREEL;
                       INTREEL[I,INDMAX+1]:=RES
                     END
            END
        END
        ELSE BEGIN
               INTREEL[I,INDMAX+1]:=INTREEL[I,INDMAX+1]
                                    -(INTREEL[I,INDICE]*XR[INDICE].SOLUTION);
               INTREEL[I,INDICE]:=0
             END
END;
(* ------------------------------ *)
PROCEDURE CHERCHERPIVOT (LIGNE:INTEGER;
                         VAR INDICE,posindice:INTEGER;
                         VAR INDIC:BOOLEAN);
(* selectionne la variable a eliminer a l'etape suivante *)
VAR I,J,MIN,NBCOEF,LNONNUL : INTEGER;
    TEST:REAL;
    TROUVE : BOOLEAN;
BEGIN
  MIN:=NBLIGNE+1;  (* +1 PCQ TTES LES LIGNES PEUVENT CONTENIR nbligne VAR
  NBCOEF:=0;INDIC:=FALSE;
  FOR J:=NBXELIM+1 TO (INDMAX-NCOLVIDE)
   DO BEGIN
        FOR I:=LIGNE TO NBLIGNE-NLIGVIDE
         DO BEGIN
              IF MODE=0 THEN TEST:=INTER[I,VIND[J]].NUM
                        ELSE TEST:=INTREEL[I,VIND[J]];
              IF TEST<>0
               THEN NBCOEF:=NBCOEF+1
            END;
        IF NBCOEF<MIN
         THEN BEGIN
                MIN:=NBCOEF;
                INDICE:=VIND[J];
                POSINDICE:=J
              END;
        NBCOEF:=0;
      END;
  IF MODE=0 THEN TEST:=INTER[LIGNE,INDICE].NUM
            ELSE TEST:=INTREEL[LIGNE,INDICE];
  IF TEST=0
   THEN BEGIN
          I:=LIGNE+1;
          TROUVE:=FALSE;
          WHILE (I<=(NBLIGNE-NLIGVIDE)) AND (NOT TROUVE)
          DO BEGIN
               IF MODE=0 THEN TEST:=INTER[I,INDICE].NUM
                         ELSE TEST:=INTREEL[I,INDICE];
               IF TEST<>0
                THEN BEGIN
                       TROUVE:=TRUE;
                       LNONNUL:=I
                     END
                ELSE I:=I+1
             END;
          ECHANGERLIGNE(LIGNE,LNONNUL);
        END;
    IF MIN=1 THEN INDIC:=TRUE;
END;
```

```
(* ------------------------------ *)
PROCEDURE ELIMPIVOT(LIGNE,INDICE:INTEGER);
(* realise une etape de l'elimination sur base du pivot selectionne par
   CHERCHERPIVOT *)
VAR I,J:INTEGER;
    MUL1,MUL2,TAMP1,TAMP2 : FRACTION;
    RESUL,TEST,MULR1,MULR2:REAL;
    CODEERREUR:ZEROUN;
BEGIN
  IF MODE=0
   THEN BEGIN
          MUL2.NUM:=INTER[LIGNE,INDICE].NUM;
          MUL2.DEN:=INTER[LIGNE,INDICE].DEN
        END
   ELSE MULR2:=INTREEL[LIGNE,INDICE];
  FOR I:=LIGNE+1 TO NBLIGNE-NLIGVIDE
   DO BEGIN
        IF MODE=0
         THEN BEGIN
                MUL1.NUM:=INTER[I,INDICE].NUM;
                MUL1.DEN:=INTER[I,INDICE].DEN;
                TEST:=INTER[I,INDICE].NUM;
              END
         ELSE BEGIN
                MULR1:=INTREEL[I,INDICE];
                TEST:=INTREEL[I,INDICE]
              END;
        IF TEST<>0
         THEN
          FOR J:=NBXELIM+1 TO INDMAX+1
           DO IF MODE=0
               THEN
                BEGIN
                  MULTIPLICATION(MUL1.NUM,MUL1.DEN,INTER[LIGNE,VIND[J]].NUM,
                                 INTER[LIGNE,VIND[J]].DEN,TAMP1.NUM,TAMP1.DE
                                 RESUL,CODEERREUR);
                 IF CODEERREUR=0
                  THEN
                   BEGIN
                     MULTIPLICATION(MUL2.NUM,MUL2.DEN,INTER[I,VIND[J]].NUM,
                                    INTER[I,VIND[J]].DEN,TAMP2.NUM,TAMP2.DE
                                    RESUL,CODEERREUR);
                    IF CODEERREUR=0
                     THEN
                      BEGIN
                        SOUSTRACTION(TAMP1.NUM,TAMP1.DEN,TAMP2.NUM,TAMP2.D
                                     INTER[I,VIND[J]].NUM,INTER[I,VIND[J]]
                                     RESUL,CODEERREUR);
                       IF CODEERREUR=1
                        THEN
                         BEGIN
                           MODE:=1;PASSAGEREEL;
                           INTREEL[I,VIND[J]]:=RESUL;
                           MULR1:=MUL1.NUM/MUL1.DEN;
                           MULR2:=MUL2.NUM/MUL2.DEN
                         END
                      END
                    ELSE
                     BEGIN
                       MODE:=1;PASSAGEREEL;
                       RESUL:=(TAMP1.NUM/TAMP1.DEN)-RESUL;
                       INTREEL[I,VIND[J]]:=RESUL;
                       MULR1:=MUL1.NUM/MUL1.DEN;
                       MULR2:=MUL2.NUM/MUL2.DEN
                     END
```

```
                        END
                      ELSE
                        BEGIN
                          MODE:=1;PASSAGEREEL;
                          RESUL:=RESUL-((MUL2.NUM/MUL2.DEN)
                                *(INTER[I,VIND[J]].NUM/INTER[I,VIND[J]].DEN))
                          INTREEL[I,VIND[J]]:=RESUL;
                          MULR1:=MUL1.NUM/MUL1.DEN;
                          MULR2:=MUL2.NUM/MUL2.DEN
                        END
                    END
                  ELSE INTREEL[I,VIND[J]]:=(INTREEL[LIGNE,VIND[J]]*MULR1)
                                          -(INTREEL[I,VIND[J]]*MULR2)
        END
    END;
(* ---------------------------------- *)
PROCEDURE INIRESOLUTION;
(* initialisation avant d'effectuer la resolution
     - initialiser lg vecteur d'indirection
     - initialiser les valeurs des solutions
     - positionner les differentes variables
     - chercher les colonnes vides de depart *)
BEGIN
  FOR I:=1 TO INDMAX+1  (* INITIALISATION *)
   DO VIND[I]:=I;      (* DU VECTEUR D'INDIRECTION *)
  FOR I:=1 TO INDMAX
   DO BEGIN
        XF[I].VALEUR:=FALSE;
        XR[I].VALEUR:=FALSE;
      END;
  POSLIGNE:=1;POSVAR:=1;
  NBXVAL:=0;NBXELIM:=0;
  NBETAPE:=NBLIGNE;ETAPE:=1;
  NLIGVIDE:=0;NCOLVIDE:=0;
  IMPOSSIBLE:=FALSE;
  COLVIDE(CVIDE,NROCVIDE,POSITION);
   WHILE CVIDE
    DO BEGIN
         FININD(NROCVIDE,POSITION);
         NCOLVIDE:=NCOLVIDE+1;
         COLVIDE(CVIDE,NROCVIDE,POSITION);
       END;
END;

PROCEDURE INTERPRETATION;
(* realise l'interpretation des resultats a partir de la matrice
   transformee par les eliminations succesives *)
VAR QCQ:ARRAY [1..DIML] OF INTEGER;
    I,J:INTEGER;
(* ---------------------------------- *)
PROCEDURE XVALEUR (VAR TROUVE,IMPOSSIBLE:BOOLEAN;
                   VAR LIGNE,INDICE:INTEGER);
(* meme genre de procedure que YATILXVALEUR mais par exple,l'ordre de
   prise en consideration des lignes est inverse *)
VAR I,J,NBCOEF:INTEGER;
    TEST : REAL ;
BEGIN
  INDICE:=0;LIGNE:=0;
  TROUVE:=FALSE;
  I:=(NBLIGNE-NLIGVIDE);NBCOEF:=0;
  WHILE (I>NBXVAL)
    AND (NOT TROUVE)
    AND (NOT IMPOSSIBLE)
    DO
     BEGIN
```

```
      FOR J:=1 TO INDMAX
       DO BEGIN
            IF MODE=0 THEN TEST:=INTER[I,J].NUM
                      ELSE TEST:=INTREEL[I,J];
            IF TEST <> 0
             THEN BEGIN
                    NBCOEF:=NBCOEF+1;
                    INDICE:=J
                  END
          END;
        IF NBCOEF=0
         THEN BEGIN
                INDICE:=0;
                IF MODE=0 THEN TEST:=INTER[I,INDMAX+1].NUM
                          ELSE TEST:=INTREEL[I,INDMAX+1];
                IF TEST = 0
                 THEN BEGIN
                        ECHANGERLIGNE(I,(NBLIGNE-NLIGVIDE));
                        NLIGVIDE:=NLIGVIDE+1;
                        I:=I-1
                      END
                 ELSE BEGIN
                        IMPOSSIBLE:=TRUE;
                        SOLIMPOS;
                      END
              END
         ELSE
          IF NBCOEF=1
           THEN BEGIN
                  TROUVE:=TRUE;LIGNE:=I
                END
           ELSE BEGIN
                  INDICE:=0;NBCOEF:=0;
                  I:=I-1
                END
      END
END;
(* ---------------------------------- *)
PROCEDURE SOLUNIQUE;
(* presente les valeurs des differentes variables en cas de solution unique
VAR I:INTEGER;
BEGIN
  TYPESOL:=UNIQUE;
  IF PRESENTATION=1
  THEN
  BEGIN
    BIP;PAGE(OUTPUT);WRITELN;
    WRITELN('****************************************');
    WRITELN('* CE SYSTEME ADMET UNE SEULE SOLUTION  *');
    WRITELN('*                                      *');
    FOR I:=1 TO INDMAX
    DO IF MODE=0
        THEN
         IF (XF[I].NUM=0) OR (XF[I].DEN=1)
          THEN BEGIN
                 GOTOXY(0,3+I);WRITELN ('* X ',I,' = ',XF[I].NUM);
                 GOTOXY(39,3+I);WRITELN('*');
               END
          ELSE BEGIN
                 GOTOXY(0,3+I);WRITELN ('* X ',I,' = ',XF[I].NUM,'/',XF[I].
                 GOTOXY(39,3+I);WRITELN('*');
               END
        ELSE BEGIN
               GOTOXY(0,3+I);WRITELN('* X ',I,' = ',XR[I].SOLUTION);
               GOTOXY(39,3+I);WRITELN('*');
```

```
            END;
    WRITELN('*                                                *');
    WRITELN('**********************************************');
    SUITE;
    (*IF (INDCHEMIN=0) AND (INDGRAPH=1)
     THEN GRAPHIQUE(MAT,MREEL);*)
  END
END;
(* ---------------------------------- *)
PROCEDURE SOLINDETERMINEE;
(*presente la solution indeterminee *)
VAR INDICE,NROLIGNE,I,J:INTEGER;
    TEST:REAL;
    VU:ARRAY [1..DIML] OF INTEGER;
(* ------------------------------ *)
PROCEDURE EXPRIMERX(LIGNE,INDICE:INTEGER);
(* exprime la valeur d'un "x" indetermine *)
VAR J:INTEGER;
    FRAC,SIGNE : STRING ;
BEGIN
  WRITE('* X ',INDICE,' =( ');
  IF MODE=0
   THEN BEGIN
          IF INTER[LIGNE,INDMAX+1].NUM<>0
           THEN
            IF INTER[LIGNE,INDMAX+1].DEN=1
             THEN WRITE(INTER[LIGNE,INDMAX+1].NUM)
             ELSE WRITE(INTER[LIGNE,INDMAX+1].NUM,
                        '/',INTER[LIGNE,INDMAX+1].DEN);
          FOR J:=1 TO INDMAX
           DO IF (INTER[LIGNE,J].NUM<>0)
              AND (J<>INDICE)
               THEN
                BEGIN
                  IF INTER[LIGNE,J].NUM<0
                   THEN SIGNE:='+'
                   ELSE SIGNE:='-';
                  DENOM(INTER[LIGNE,J].NUM,INTER[LIGNE,J].DEN,FRAC);
                  WRITE(SIGNE,FRAC,' X',J)
                END;
          IF INTER[LIGNE,INDICE].DEN=1
           THEN
            IF INTER[LIGNE,INDICE].NUM=1
             THEN WRITELN(' )')
             ELSE WRITELN(' )/',INTER[LIGNE,INDICE].NUM)
           ELSE
            BEGIN
              DENOM(INTER[LIGNE,INDICE].DEN,
                    INTER[LIGNE,INDICE].NUM,
                    FRAC);
              WRITELN(' ) * ',FRAC)
            END
        END
   ELSE BEGIN
          IF (INTREEL[LIGNE,INDMAX+1]<>0)
           THEN WRITE (INTREEL[LIGNE,INDMAX+1]);
          FOR J:=1 TO INDMAX
           DO IF (INTREEL[LIGNE,J]<>0)
              AND (INDICE<>J)
               THEN BEGIN
                      IF INTREEL[LIGNE,J]<0
                       THEN SIGNE:='+'
                       ELSE SIGNE:='-';
                      WRITE(SIGNE,ABS(INTREEL[LIGNE,J]),' X',J);
                    END;
```

```
          IF INTREEL[LIGNE,INDICE]<>1
           THEN WRITELN(' )/ ',INTREEL[LIGNE,INDICE])
           ELSE WRITELN(' )')
        END
END;
(* ---------------------------- *)
FUNCTION DEJAVU(X:INTEGER):BOOLEAN;
(* fonction vraie si la variable d'indice "x" appartient au vecteur "vu"
   cad si cette var a deja ete exprimee ou si elle a une valeur *)
VAR I:INTEGER;
BEGIN
  DEJAVU:=FALSE;
  FOR I:=1 TO INDICE
   DO IF X=VU[I] THEN DEJAVU:=TRUE
END;
(* ---------------------------- *)
BEGIN (* DE LA PROCEDURE SOLINDETERMINEE *)
  TYPESOL:=INDET;
  IF PRESENTATION=1 THEN
  BEGIN
    BIP;PAGE(OUTPUT);WRITELN;
    WRITELN('*****************************************');;
    WRITELN('* CE SYSTEME EST INDETERMINE :          *');
    WRITELN('* L''ORDRE D''INDETERMINATION = ',ORDREIND,'        *');
    WRITELN('*                                       *');
    FOR I:=1 TO ORDREIND
     DO BEGIN
          GOTOXY(0,4+I);WRITELN ('* X ',QCQ[I],' EST QUELCONQUE');
          GOTOXY(39,4+I);WRITELN('*');
        END;
    NROLIGNE:=0;
    FOR I:=1 TO INDMAX
     DO
      IF(MODE=0) AND (XF[I].VALEUR)
         THEN BEGIN
                NROLIGNE:=NROLIGNE+1;
                GOTOXY(0,4+ORDREIND+NROLIGNE);
                IF (XF[I].NUM=0) OR (XF[I].DEN=1)
                 THEN WRITELN ('* X ',I,' = ',XF[I].NUM)
                 ELSE WRITELN ('* X ',I,' = ',XF[I].NUM,'/',XF[I].DEN);
                GOTOXY(39,4+ORDREIND+NROLIGNE);
                WRITELN('*');
              END
         ELSE
          IF (MODE=1) AND (XR[I].VALEUR)
           THEN BEGIN
                  NROLIGNE:=NROLIGNE+1;
                  GOTOXY(0,4+ORDREIND+NROLIGNE);
                  WRITELN ('* X ',I,' = ',XR[I].SOLUTION);
                  GOTOXY(39,4+ORDREIND+NROLIGNE);
                END;
    FOR I:=1 TO ORDREIND
     DO VU[I]:=QCQ[I];
    INDICE:=ORDREIND;
    FOR I:=(NBLIGNE-NLIGVIDE) DOWNTO NBXVAL+1
     DO BEGIN
          J:=1;
          TROUVE:=FALSE;
          WHILE (NOT TROUVE)
           DO BEGIN
                IF MODE=0 THEN TEST:=INTER[I,J].NUM
                          ELSE TEST:=INTREEL[I,J];
                IF (TEST<>0) AND (NOT DEJAVU(J))
                 THEN BEGIN
                        TROUVE:=TRUE;
```

```
                        NROLIGNE:=NROLIGNE+1;
                        GOTOXY(0,4+ORDREIND+NROLIGNE);
                        EXPRIMERX(I,J);
                        GOTOXY(39,4+ORDREIND+NROLIGNE);
                        WRITELN('*');
                        INDICE:=INDICE+1;
                        VU[INDICE]:=J
                      END
                  ELSE J:=J+1
              END
        END;
    WRITELN('****************************************');
    SUITE;
    (*IF (INDCHEMIN=0) AND (INDGRAPH=1)
     THEN GRAPHIQUE(MAT,MREEL);*)
  END;
END;
(* ---------------------------------- *)
BEGIN (* DE LA PROCEDURE INTERPRETATION *)
  POSLIGNE:=NBXVAL+1;
  XVALEUR(TROUVE,IMPOSSIBLE,LIGNE,INDICE);
  WHILE TROUVE AND NOT IMPOSSIBLE
   DO BEGIN
        NBXVAL:=NBXVAL+1;
        DONNERVALEUR(LIGNE,INDICE);
        CHANGERORDRE(LIGNE,INDICE);
        INJECTER(INDICE,NBXVAL);
        POSLIGNE:=POSLIGNE+1;
        IF INDCHEMIN=1
         THEN BEGIN
                PAGE(OUTPUT);GOTOXY(0,0);
                WRITELN('ON A REMPLACE X ',INDICE,' PAR SA VALEUR');
                WRITELN('********************************');
                IMPMAT(INTER,INTREEL);
                SUITE;
                (*IF INDGRAPH=1
                 THEN GRAPHIQUE(INTER,INTREEL);*)
              END;
        XVALEUR(TROUVE,IMPOSSIBLE,LIGNE,INDICE)
      END;
  IF NOT IMPOSSIBLE
   THEN
    IF INDMAX=NBXVAL
     THEN SOLUNIQUE
     ELSE BEGIN
            ORDREIND:=INDMAX-(NBLIGNE-NLIGVIDE);
            I:=1;
            FOR J:=INDMAX DOWNTO INDMAX-ORDREIND+1
             DO BEGIN QCQ[I]:=VIND[J];
                      I:=I+1
                END;
            SOLINDETERMINEE;
           END;
END;
(* ---------------------------- *)
BEGIN (* DE LA PROCEDURE RESOLUTION  *)
  (*$R UTILUNIT*)
  INIRESOLUTION;
  IF INDCHEMIN=1
   THEN BEGIN
          PAGE(OUTPUT);
          WRITELN('SYSTEME DE DEPART (PEUT-ETRE SIMPLIFIE)');
          WRITELN('****************************************');
          IMPMAT(INTER,INTREEL);
          SUITE;
```

```
          (*IF INDGRAPH=1
           THEN GRAPHIQUE(INTER,INTREEL);*)
        END;
  WHILE (ETAPE<NBETAPE) AND (NOT IMPOSSIBLE)
   DO
    BEGIN
      COLVIDE(CVIDE,NROCVIDE,POSITION);
      WHILE CVIDE (* pas test etape car je peux depasser *)
       DO BEGIN
            FININD(NROCVIDE,POSITION);
            ETAPE:=ETAPE+1;
            NCOLVIDE:=NCOLVIDE+1;
            COLVIDE(CVIDE,NROCVIDE,POSITION);
          END;
      YATILXVALEUR (TROUVE,IMPOSSIBLE,LIGNE,INDICE,position);
      WHILE TROUVE AND (NOT IMPOSSIBLE) AND (ETAPE<NBETAPE)
       DO BEGIN
            NBXVAL:=NBXVAL+1;
            NBXELIM:=NBXELIM+1;
            DEBUTIND(INDICE,POSITION);
            DONNERVALEUR(LIGNE,INDICE);
            CHANGERORDRESDESLIGNES(LIGNE,INDICE);
            INJECTER(INDICE,NBXVAL);
            POSLIGNE:=POSLIGNE+1;
            IF INDCHEMIN=1
            THEN BEGIN
                   PAGE(OUTPUT);
                   GOTOXY(0,0);
                   WRITELN('ON A TROUVE LA VALEUR DE X ',INDICE);
                   WRITELN('ET ON L''A REMPLACE PAR SA VALEUR ');
                   WRITELN('**********************************');
                   IMPMAT(INTER,INTREEL);
                   SUITE;
                   (*IF INDGRAPH=1
                    THEN GRAPHIQUE(INTER,INTREEL);*)
                 END;
            YATILXVALEUR(TROUVE,IMPOSSIBLE,LIGNE,INDICE,position);
            ETAPE:=ETAPE+1;
          END;
      IF (ETAPE<NBETAPE) AND (NOT IMPOSSIBLE)
       THEN BEGIN
              CHERCHERPIVOT(POSLIGNE,POSVAR,position,RIENAELIMINER);
              IF RIENAELIMINER
               THEN
               ELSE ELIMPIVOT(POSLIGNE,POSVAR);
              NBXELIM:=NBXELIM+1;
              DEBUTIND(POSVAR,POSITION);
              IF INDCHEMIN=1
               THEN BEGIN
                      PAGE(OUTPUT);GOTOXY(0,0);
                      WRITELN ('ON A ELIMINE X ',POSVAR);
                      WRITELN ('*****************');
                      IMPMAT(INTER,INTREEL);
                      SUITE;
                      (*IF INDGRAPH=1
                       THEN GRAPHIQUE(INTER,INTREEL);*)
                    END;
              ETAPE:=ETAPE+1;
              POSLIGNE:=POSLIGNE+1
            END;
      END;
  IF NOT IMPOSSIBLE THEN INTERPRETATION;
END;
(* ---------------------------- *)
BEGIN END.
```

```
****************************************
*                                      *
*  U N I T S    G R A P H I Q U E S    *
*                                      *
****************************************
```

* DESSIN22 : ELLE COMPREND LES DECLARATIONS DE DONNEES ET LES PROCEDURES UTILISEES POUR LA REPRESENTATION DES DROITES ET DES SYSTEME 2*2.
(voir pp 43-52)

* DECLAR33 : ELLE COMPREND LES DECLARATIONS DE DONNEES UTILISEES POUR LES REPRESENTATIONS DE PLANS ET DE SYSTEMES 3*3.
(voir pp 53-54)

* CHOIXPLAN : ELLE COMPREND LES PROCEDURES DE CHOIX DES SOMMETS ET DES ORIENTATIONS.
(voir pp 55-61)

* INTER33 : ELLE COMPREND LES PROCEDURES QUI CALCULENT LES INTERSECTION ENTRE 2 PLANS.
(voir pp 62-72)

* DESSIN33 : ELLE COMPREND LES PROCEDURES QUI PERMETTENT LA REPRESENTATION DES PLANS ET DES SYSTEMES 3*3.
(voir pp 73-81)

```
(*$S+,N+*)
UNIT DESSIN22;INTRINSIC CODE 26 DATA 27;

INTERFACE
(*======*)
USES APPLESTUFF,UTILUNIT,TURTLEGRAPHICS;

CONST CENTREX = 139;
      CENTREY = 95;
      LONGUEUR = 280;
      LARGEUR = 192;

TYPE EXTREM = RECORD
        XX1,YY1,XX2,YY2:REAL
              END;

     QUELINTER=(CONFONDU,PARALLELE,SECANT);

VAR NDR:INTEGER; (* NDR = Numero de DRoite *)

    TYPEINTER:QUELINTER;

    X1,Y1,X2,Y2,INTX,INTY,E1,E2,NX,NY,UN:REAL;(* UN = UNite *)

    PASDROITE,RECALC:BOOLEAN;

    P:ARRAY[1..2] OF EXTREM; (* P=tableau de Points *)

    TABUNI:ARRAY [1..2] OF REAL; (* TABUNI = TABleau d'UNIte *)

    UNITEINTER:REAL;(* UNITE QUI PERMET DE VOIR L'INTERSECTION *)

    DIND,DIMP:ARRAY[1..2]OF BOOLEAN;(* DIND = Droite INDetermine
                                       DIMP = Droite IMPossible
                                       CES 2 TABLEAUX MEMORISENT SI 1 OU 2
                                       EQUATIONS SONT SOIT IMPOS.,SOIT INDET


PROCEDURE GR22(AF,BF,CF:FRACTION;
               AR,BR,CR:REAL);
PROCEDURE SYST22(MATF:MATRICE;
                 MATR:MATREEL);

(* ---------------------------------- *)

IMPLEMENTATION
(*==========*)
 PROCEDURE C0;(* METTRE COULEUR A RIEN *)
  BEGIN PENCOLOR(NONE) END;
 (* ---------------------------------- *)
 PROCEDURE C1;(* METTRE COULEUR A BLANC *)
  BEGIN PENCOLOR(WHITE) END;
 (* ---------------------------------- *)
 PROCEDURE AXES;
  (* TRACE LES AXES ET LES NOMME (X1,X2) *)
  BEGIN
    (*$R TURTLEGRAPHICS*)
    INITTURTLE;
    C0;
    MOVETO(CENTREX,0);
    C1;
    MOVETO(CENTREX,191);
    C0;
    MOVETO(0,CENTREY);
    C1;
```

```
    MOVETO(279,CENTREY);
    CO;
    MOVETO(264,CENTREY-10);
    C1;
    WSTRING('X1');
    CO;
    MOVETO(CENTREX-16,0);
    C1;
    WSTRING('X2');
  END;
(* ------------------------------- *)

 PROCEDURE DROITE(X1,Y1,X2,Y2,UN:REAL);
  (* TRACE UN SEGMENT DE DROITE A L'ECRAN ENTRE LES POINTS (X1,Y1) ET
     (X2,Y2) EN FONCTION DE L'UNITE "UN" *)

  VAR TAMPON1,TAMPON2:INTEGER;

  BEGIN
    (*$R TURTLEGRAPHICS*)
    IF (X1<>0) OR (Y1<>0) OR (X2<>0) OR (Y2<>0)
    THEN
     BEGIN
       CO;
       TAMPON1:=ROUND(CENTREX+(X1*UN));
       TAMPON2:=ROUND(CENTREY+(Y1*UN));
       MOVETO(TAMPON1,TAMPON2);
       C1;
       TAMPON1:=ROUND(CENTREX+(X2*UN));
       TAMPON2:=ROUND(CENTREY+(Y2*UN));
       MOVETO(TAMPON1,TAMPON2)
     END
  END;
 (* ------------------------------ *)
  PROCEDURE DETERUNITE(INTX,INTY:REAL;
                       VAR UN:REAL);
   (* DETERMINE UNE UNITE TELLE QUE LES EVENTUELS POINTS D'INTERSECTION
      AVEC LES AXES (INTX ET INTY) SOIENT VISIBLES *)

   VAR UNITE:REAL;

   BEGIN
     UNITE:=50;
     IF INTX=0
     THEN
      IF INTY=0
      THEN
      ELSE UNITE:=ABS((LARGEUR)/(4*INTY))
     ELSE
      BEGIN
        UNITE:=ABS((LONGUEUR)/(4*INTX));
        IF ABS(INTY*UNITE)>LARGEUR/2
         THEN UNITE:=ABS((LARGEUR)/(4*INTY))
      END;
     UN:=UNITE;
   END;
   (* ---------------------------- *)
   PROCEDURE CHOIXUNITE(AF,BF,CF:FRACTION;
                        AR,BR,CR:REAL);
   (* DETERMINE AVEC QUELS PARAMETRES IL FAUT APPELER DETERUNITE *)
   VAR A,B,C,TAMPON1,TAMPON2:REAL;

   BEGIN
     IF MODE=0
     THEN
```

```
    BEGIN
      A:=AF.NUM;
      B:=BF.NUM;
      C:=CF.NUM;
      IF A<>0
      THEN
       BEGIN
         TAMPON1:=C*AF.DEN;
         TAMPON1:=TAMPON1/(AF.NUM*CF.DEN);
       END
      ELSE TAMPON1:=0;
      IF B<>0
      THEN
       BEGIN
         TAMPON2:=C*BF.DEN;
         TAMPON2:=TAMPON2/(BF.NUM*CF.DEN);
       END
      ELSE TAMPON2:=0
    END
    ELSE
    BEGIN
      A:=AR;
      B:=BR;
      C:=CR;
      IF A<>0 THEN TAMPON1:=CR/AR
              ELSE TAMPON1:=0;
      IF B<>0 THEN TAMPON2:=CR/BR
              ELSE TAMPON2:=0;
    END;
    DETERUNITE(TAMPON1,TAMPON2,UN);
  END;
(* ---------------------------- *)
  PROCEDURE CALCULPTS(AF,BF,CF:FRACTION;
                      AR,BR,CR:REAL);
  (* CALCUL DES EXTREMITES DU SEGMENT ET ENVOI DE MESSAGES EXPLICATIFS *)

  VAR A,B,C:REAL;
  (* ------------------------- *)

   (* ---------------------------- *)
   PROCEDURE DRINDET;
      (* MESSAGE EN CAS D'EQUATION INDETERMINEE *)
      BEGIN
        PAGE(OUTPUT);
        GOTOXY(0,10);
        TEXTMODE;
        PASDROITE:=TRUE;
        WRITELN ('CETTE EQUATION EST COMPLETEMENT');
        WRITELN ('INDETERMINEE.TOUTES LES VALEURS');
        WRITELN ('DE X1 ET DE X2 CONVIENNENT PUISQUE');
        WRITELN ('0X1+0X2=0 EST TOUJOURS VRAI.');
     END;
  (* ------------------------ *)

   PROCEDURE DRIMPOS;
      (* MESSAGE EN CAS D'EQUATION IMPOSSIBLE *)
      BEGIN
        PAGE(OUTPUT);
        GOTOXY(0,10);
        TEXTMODE;
        PASDROITE:=TRUE;
        WRITELN ('CETTE EQUATION EST IMPOSSIBLE A REPRE-');
        WRITELN ('SENTER CAR AUCUNE VALEUR DE X1 ET X2');
        WRITELN ('NE CONVIENT PUISQUE 0X1+0X2<>0 EST');
        WRITELN ('TOUJOURS IMPOSSIBLE.');
```

```
   END;
(* ------------------------- *)
 PROCEDURE DRPARX;
   (* MESSAGE EN CAS DE DROITE PARALLELE A L'AXE X1 *)
    BEGIN
     PAGE(OUTPUT);
     GOTOXY(0,10);
     TEXTMODE;
     IF ((MODE=0) AND (CF.NUM=0))
     OR ((MODE=1) AND (cR=0))
      THEN WRITELN('CETTE DROITE EST CONFONDUE AVEC L''AXE X1.')
      ELSE
      BEGIN
        WRITELN('CETTE DROITE EST PARALLELE A L''AXE X1.');
        WRITE('ELLE COUPE L''AXE X2 AU POINT ');
        IF MODE=0
         THEN WRITELN((CF.NUM/CF.DEN)/(BF.NUM/BF.DEN))
         ELSE WRITELN(CR/BR)
      END;
     SUITE;
   END;
(* ------------------------- *)
  PROCEDURE DRPARY;
   (* MESSAGE EN CAS DE DROITE PARALLELE A L'AXE X2 *)
    BEGIN
     PAGE(OUTPUT);
     GOTOXY(0,10);
     TEXTMODE;
     IF ((MODE=0) AND (CF.NUM=0))
     OR ((MODE=1) AND (CR=0))
      THEN WRITELN('CETTE DROITE EST CONFONDUE AVEC L''AXE X2.')
      ELSE
      BEGIN
        WRITELN('CETTE DROITE EST PARALLELE A L''AXE X2.');
        WRITE('ELLE COUPE L''AXE X1 AU POINT ');
        IF MODE=0
        THEN WRITELN((CF.NUM/CF.DEN)/(AF.NUM/AF.DEN))
        ELSE WRITELN(CR/AR)
      END;
     SUITE;
   END;
(* ------------------------- *)
  PROCEDURE DRPPOO;
   VAR F:FRACTION;
   (* MESSAGE EN CAS DE DROITE PASSANT PAR L'ORIGINE *)
    BEGIN
     PAGE(OUTPUT);
     GOTOXY(0,10);
     TEXTMODE;
     WRITELN('CETTE DROITE PASSE PAR LE POINT ORIGINE.');
     WRITE('SA PENTE VAUT : ');
     IF MODE=0
     THEN WRITELN((-BF.NUM/BF.DEN)/(AF.NUM/AF.DEN))
     ELSE WRITELN(-BR/AR);
     WRITELN('');
     WRITELN('LA DISTANCE ENTRE L''ORIGINE ET LES');
     WRITELN('POINTS E1 ET E2 REPRESENTE 1 UNITE.');
     SUITE;
   END;
(* ------------------------- *)

 PROCEDURE DRPP2P(A,B,C:REAL);
   (* MESSAGE EN CAS DE DROITE SECANTE AUX DEUX AXES *)
    BEGIN
     PAGE(OUTPUT);
```

```
      GOTOXY(0,10);
      TEXTMODE;
      WRITELN('CETTE DROITE EST SECANTE AUX DEUX AXES.');
      WRITE('ELLE COUPE L''AXE X1 AU POINT E1=');
      IF MODE=0
       THEN WRITELN((CF.NUM/CF.DEN)/(AF.NUM/AF.DEN))
       ELSE WRITELN(CR/AR);
      WRITE('ET ELLE COUPE L''AXE X2 AU POINT E2=');
      IF MODE=0
       THEN WRITELN((CF.NUM/CF.DEN)/(BF.NUM/BF.DEN))
       ELSE WRITELN(C/B);
      SUITE;
    END;
(* ------------------------- *)


BEGIN
 IF MODE=0
  THEN BEGIN
         A:=AF.NUM/AF.DEN;
         B:=BF.NUM/BF.DEN;
         C:=CF.NUM/BF.DEN
       END
  ELSE BEGIN
         A:=AR;
         B:=BR;
         C:=CR
       END;
 IF A=0
  THEN
   IF B=0
    THEN
     IF RECALC (* SI RECALC EST VRAI CELA SIGNIFIE QUE NOUS SOMMES EN
                   PHASE DE RECALCUL PCQ L'UNITE A ETE MODIFIEE:DONC
                   NOUS N'ENVOYONS PLUS LES MESSAGES *)

      THEN
      ELSE
       IF C=0
        THEN BEGIN DIND[NDR]:=TRUE;
                   DRINDET
             END
        ELSE BEGIN DIMP[NDR]:=TRUE;
                   DRIMPOS
             END
     ELSE
      BEGIN (* BY=C -> AXE // A L'HORIZONTAL*)
        E1:=0;E2:=C/B;
        X1:=-LONGUEUR/(2*UN);Y1:=E2;
        X2:=LONGUEUR/(2*UN);Y2:=Y1;
        IF NOT RECALC
         THEN DRPARX;
      END
    ELSE
     IF B=0
     THEN
      BEGIN (* AX=C -> AXE // A LA VERTICALE*)
        E1:=C/A;E2:=0;
        X1:=E1;Y1:=-LARGEUR/(2*UN);
        X2:=X1;Y2:=LARGEUR/(2*UN);
        IF NOT RECALC
         THEN DRPARY;
      END
     ELSE
      IF C=0
```

```
          THEN
           BEGIN
             E1:=1;E2:=1;
             IF NOT RECALC
             THEN DRPPOO;
             IF ABS(A*LONGUEUR)>ABS(B*LARGEUR)
             THEN
              BEGIN
                Y1:=-LARGEUR/(2*UN);X1:=-(B/A)*Y1;
                Y2:=-Y1;X2:=-X1
              END
             ELSE
              BEGIN
                X1:=-LONGUEUR/(2*UN);Y1:=-(A/B)*X1;
                X2:=-X1;Y2:=-Y1
              END
           END
          ELSE
           BEGIN
             E1:=C/A;E2:=C/B;
             IF NOT RECALC
             THEN DRPP2P(A,B,C);
             NX:=LONGUEUR/(2*UN);NY:=LARGEUR/(2*UN);
             X1:=-NX;Y1:=(C-(A*X1))/B;
             IF ABS(Y1)<=ABS(NY)
             THEN
              BEGIN
                X2:=NX;Y2:=(C-(A*X2))/B;
                IF ABS(Y2)<=ABS(NY)
                 THEN  (* DROITE DU BORD GAUCHE AU BORD DROIT  *)
                 ELSE
                  BEGIN
                    Y2:=NY;X2:=(C-(B*Y2))/A;
                    IF ABS(X2)<=ABS(NX)
                    THEN (* DROITE DU BORD GAUCHE AU BORD SUPERIEUR *)
                    ELSE
                     BEGIN (* DROITE DU BORD GAUCHE AU BORD INFERIEUR *)
                       Y2:=-NY;X2:=(C-(B*Y2))/A;
                     END
                  END
              END
             ELSE
              BEGIN
                X1:=NX;Y1:=(C-(A*X1))/B;
                IF ABS(Y1)<=ABS(NY)
                THEN
                 BEGIN
                   Y2:=-NY;X2:=(C-(B*Y2))/A;
                   IF ABS(X2)<=ABS(NX)
                   THEN (* DROITE DU BORD DROIT AU BORD INFERIEUR *)
                   ELSE
                    BEGIN (* DROITE DU BORD DROIT AU BORD SUPERIEUR *)
                      Y2:=NY;X2:=(C-(B*Y2))/A;
                    END
                 END
                ELSE
                 BEGIN (* DROITE DU BORD SUPERIEUR AU BORD INFERIEUR *)
                   Y1:=NY;X1:=(C-(B*Y1))/A;
                   Y2:=-NY;X2:=(C-(B*Y2))/A;
                 END
              END
           END
      END;
(* ------------------------------ *)
PROCEDURE GR22;
```

```
(* RAPPEL DES PARAMETRES :(AF,BF,CF:FRACTION;
                          AR,BR,CR:REAL);*)
(* ELLE PERMET LA REPRESENTATION D'UNE EQUATION A 2 INCONNUES ;
   SELON LE MODE ON LUI PASSE 3 FRACTIONS OU 3 REELS *)

   (* ---------------------------- *)
PROCEDURE INTERSECTION(INTX,INTY:REAL);
(* ELLE SITUE LE OU LES POINTS D'INTERSECTION AVEC LES AXES PAR UN '+'
   ET POSITIONNE 'E1' ET 'E2' SUR LE SCHEMA COMME POINTS DE REPERE *)

VAR TAMPON:INTEGER;

BEGIN
   (*$R TURTLEGRAPHICS*)
   IF INTX<>0
   THEN
    BEGIN
      CO;
      TAMPON:=ROUND(CENTREX+(UN*INTX));
      MOVETO(TAMPON-3,CENTREY-4);
      WSTRING('+');
      MOVETO(tampon-9,CENTREY-10);
      WSTRING('E1');
    END;
   IF INTY<>0
   THEN
    BEGIN
      CO;
      TAMPON:=ROUND(CENTREY+(UN*INTY));
      MOVETO(CENTREX-3,TAMPON-4);
      WSTRING('+');
      MOVETO(CENTREX-16,tampon-4);
      WSTRING('E2');
    END
END;
(* ------------------------------ *)
PROCEDURE ECHELLE(UN:REAL);
(* ELLE EXPRIME L'ECHELLE UTILISEE POUR LE DESSIN COURANT *)

VAR TAMPON,NUM,DEN:INTEGER;
    CHAINE:STRING;

BEGIN
   (*$R TURTLEGRAPHICS*)
   IF UN>1
    THEN
     BEGIN
         NUM:=ROUND(UN);
         DEN:=1
     END
    ELSE
     BEGIN
       NUM:=1;
       DEN:=ROUND(1/UN)
     END;
   CO;
   MOVETO(0,LARGEUR-10);
   C1;
   WSTRING('ECH:');
   STR(NUM,CHAINE);
   WSTRING(CHAINE);
   WSTRING('/');
   STR(DEN,CHAINE);
   WSTRING(CHAINE);
END;
```

```
  (* ---------------------------- *)

 BEGIN
   PASDROITE:=FALSE; (* PASDROITE SERA VRAI SI NOUS NE REPRESENTONS
                        PAS DE DROITE, CAD EQUATION IMPOS OU INDET *)

   CHOIXUNITE(AF,BF,CF,AR,BR,CR);
   CALCULPTS(AF,BF,CF,AR,BR,CR);
   IF PASDROITE
    THEN
    ELSE BEGIN
           AXES;
           DROITE(X1,Y1,X2,Y2,UN);
           TABUNI[NDR]:=UN;
           WITH P[NDR]
           DO
            BEGIN
              XX1:=X1;XX2:=X2;
              YY1:=Y1;YY2:=Y2
            END;
           INTERSECTION(E1,E2);
           ECHELLE(UN);
         END;
   END;

(* ---------------------------------- *)
PROCEDURE MIN(N1,N2,N3:REAL;VAR MINI:REAL);
(* CHERCHE LE MINIMUM DES 3 NOMBRES REELS *)

BEGIN
  IF N1<N2
  THEN
   IF N1<N3 THEN MINI:=N1
            ELSE MINI:=N3
  ELSE
   IF N2<N3 THEN MINI:=N2
            ELSE MINI:=N3
END;
(* ---------------------------------- *)
PROCEDURE SYST22;
(* REPRESENTATION D'UN SYSTEME 2*2 *)

PROCEDURE CALCULINTER(VAR X,Y:REAL);
(* DETERMINE LA RELATION ENTRE LES 2 DROITES ET LE CAS ECHEANT CHERCHE LA
   POSITION DU POINT D'INTERSECTION *)

VAR A1,A2,B1,B2,C1,C2:REAL;
BEGIN
  IF MODE=0
   THEN BEGIN
          A1:=MATF[1,1].NUM/MATF[1,1].DEN;
          A2:=MATF[2,1].NUM/MATF[2,1].DEN;
          B1:=MATF[1,2].NUM/MATF[1,2].DEN;
          B2:=MATF[2,2].NUM/MATF[2,2].DEN;
          C1:=MATF[1,3].NUM/MATF[1,3].DEN;
          C2:=MATF[2,3].NUM/MATF[2,3].DEN;
        END
   ELSE BEGIN
          A1:=MATR[1,1];
          A2:=MATR[2,1];
          B1:=MATR[1,2];
          B2:=MATR[2,2];
          C1:=MATR[1,3];
          C2:=MATR[2,3];
```

```
        END;
  IF A1*B2=A2*B1
   THEN BEGIN X:=0;Y:=0;
              IF (A1*C2<>A2*C1) OR (B1*C2<>B2*C1)
               THEN TYPEINTER:=PARALLELE
               ELSE TYPEINTER:=CONFONDU;
        END
   ELSE BEGIN
          TYPEINTER:=SECANT;
          Y:=((a2*c1)-(a1*c2))/((a2*b1)-(a1*b2));
          IF A1=0
           THEN X:=(c2-(b2*y))/a2
           ELSE X:=(c1-(b1*y))/a1
        END
END;

(* ------------------------------- *)
PROCEDURE RECALCUL(INDICE:INTEGER);
(* RECALCULE LES POINTS EN FONCTION D'UNE NOUVELLE UNITE *)

BEGIN
  CALCULPTS(MATF[INDICE,1],MATF[INDICE,2],MATF[INDICE,3],
            MATR[INDICE,1],MATR[INDICE,2],MATR[INDICE,3]);
  P[Indice].XX1:=X1;P[Indice].XX2:=X2;
  P[Indice].YY1:=Y1;P[Indice].YY2:=Y2
END;
(* ------------------------------- *)
PROCEDURE SUITESYST22;
(* 2eme PARTIE DE SYST22 POUR QUE CETTE PROCEDURE NE SOIT PAS TROP LONGUE *

BEGIN
  IF (DIND[1]) OR (DIND[2])
  THEN BEGIN
         WRITELN('UNE DES 2 DROITES ETANT INDETERMINEE');
         WRITELN('IL Y A UNE INFINITE DE POINTS SOLUTION');
         WRITELN('ILS SONT SITUES SUR L''AUTRE DROITE.');
       END
  ELSE
  CASE TYPEINTER OF
    CONFONDU:WRITELN('CES 2 DROITES SONT CONFONDUES.');
    PARALLELE:WRITELN('CES 2 DROITES SONT PARALLELES.');
    SECANT:BEGIN
             WRITELN('CES 2 DROITES SONT SECANTES.');
             WRITELN('LEUR POINT D''INTERSECTION (X1,X2)');
             WRITE('EST (');
             IF ROUND(INTX)=INTX
              THEN WRITE(ROUND(INTX),' , ')
              ELSE WRITE(INTX,' , ');
             IF ROUND(INTY)=INTY
              THEN WRITELN(ROUND(INTY),')')
              ELSE WRITELN(INTY,')');
           END;
  END;
WRITELN('************************************');
SUITE;
MIN(TABUNI[1],TABUNI[2],UNITEINTER,UN);
IF TABUNI[1]<>UN THEN RECALCUL(1);
IF TABUNI[2]<>UN THEN RECALCUL(2);
AXES;
FOR I:=1 TO 2
DO
 DROITE(P[I].XX1,P[I].YY1,
        P[I].XX2,P[I].YY2,UN);
END;
(* ------------------------------- *)
```

```
BEGIN
  RECALC:=FALSE;
  FOR NDR:=1 TO 2
  DO
  BEGIN
    DIND[NDR]:=FALSE;
    DIMP[NDR]:=FALSE;
    WITH P[NDR]
    DO
    BEGIN
      XX1:=0;XX2:=0;YY1:=0;YY2:=0
    END;
    GR22(MATF[NDR,1],MATF[NDR,2],MATF[NDR,3],
         MATR[NDR,1],MATR[NDR,2],MATR[NDR,3]);
    SUITE;
  END;
  RECALC:=TRUE;
  CALCULINTER(INTX,INTY);
  DETERUNITE(INTX,INTY,UNITEINTER);
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('************************************');
  IF (DIMP[1]) OR (DIMP[2])
   THEN BEGIN
          WRITELN('UNE DROITE ETANT DU TYPE (0=C) ,');
          WRITELN('CE SYSTEME EST IMPOSSIBLE.');
          WRITELN('************************************');
        END
   ELSE IF (DIND[1]) AND (DIND[2])
         THEN BEGIN
                WRITELN('COMME LES 2 DROITES SONT COMPLETEMENT');
                WRITELN('INDETERMINEES,LE SYSTEME L''EST EGALEMENT.');
                WRITELN('TOUS LES POINTS SONT SOLUTION.');
                WRITELN('************************************');
              END
         ELSE SUITESYST22;
END;

(* ------------------------ *)
BEGIN END.
```

```
(*$S+*)
UNIT DECLAR33;INTRINSIC CODE 11 DATA 12;

INTERFACE
(*======*)

CONST AXEX = 140;  (* DEMI LONGUEUR DE L'ECRAN *)
      AXEY = 96;   (* DEMI LARGEUR DE L'ECRAN *)
      CX = 139;    (* MILIEU DE L'ECRAN EN LONGUEUR *)
      CY = 95;     (* MILIEU DE L'ECRAN EN LARGEUR *)
      XMAX = 279;  (* LONGUEUR DE L'ECRAN *)
      YMAX = 191;  (* LARGEUR DE L'ECRAN *)

TYPE POINT = RECORD
               P1,P2,P3:REAL
             END;
     (* permet de memoriser un point en 3 dimensions en 1 variable *)

     COORDONNEE = RECORD
                    XX:INTEGER;
                    YY:INTEGER
                  END;
     (* permet de memoriser un point en 2 dimensions en 1 variable *)

     SENS = (HORIZONTAL,VERTICAL);(* sera utile pour savoir dans quel
                                     SENS prolonger un plan *)


VAR PLAN3 : ARRAY[1..3,1..6] OF POINT;
    (* chaque plan peut etre memorise dans max 6 sommets (cas ou le plan
       est "double").PLAN3 permet de memoriser les sommets des 3 plans
       en 3 dimensions *)

    PLAN2 : ARRAY[1..3,1..6] OF COORDONNEE;
    (* meme chose que PLAN3 mais en 2 dimensions *)

    ORIENTATION : ARRAY[1..3,1..6] OF BOOLEAN;
    (* tableau qui pour chacun des 3 plans,memorise quelles sont les
       orientations permises ou non *)

    ORI:ARRAY[1..6,1..2] OF REAL;
    (*tableau qui contient les valeurs (mulx,muly) pour chaque orientatio

    INTER1,INTER2:POINT;
    (* extremites du segment d'intersection courant *)

    PTINTER:ARRAY[1..3,1..2] OF POINT;
    (* memorise les sommets des extremites des segments d'intersection
       des plans 2 a 2 *)

    NRO : INTEGER;
    NBSOMMET,TYPEDUPLAN : ARRAY[1..3] OF INTEGER;
    UNI,AA,BB,CC,DD : ARRAY[1..3] OF REAL;
    MULX,MULY,UNITE : REAL;
    CAR : CHAR;
    AX1NEG,AX2NEG,AX3NEG :ARRAY[1..3] OF BOOLEAN;
    (* tableaux permettent de determiner pour chaque axe et chaque plan
       s'il faut representer la partie negative de l'axe *)

    AXE1NEG,AXE2NEG,AXE3NEG,FIN : BOOLEAN;
    (* AXEiNEG est vrai s'il faut representer la partie negative de l'axe


PROCEDURE BIDON;
    (* ele ne contient rien mais est obligatoire pour la UNIT *)
```

```
IMPLEMENTATION
(*==========*)
PROCEDURE BIDON;
BEGIN
END;

(* CETTE UNIT NE CONTIENT QUE DES DECLARATIONS.
   AINSI LA PARTIE IMPLEMENTATION EST-ELLE VIDE. *)

BEGIN
END.
```

```
(*$S+*)
UNIT CHOIXPLAN;INTRINSIC CODE 13;
INTERFACE
(*======*)
USES DECLAR33;

PROCEDURE QUELPLAN(A,B,C,D:REAL;NRO:INTEGER);
(* ------------------------------- *)
IMPLEMENTATION
(*==========*)
PROCEDURE QUELPLAN;
(* DETERMINE PAR QUELS POINTS NOUS ALLONS REPRESENTER LE PLAN ET ETUDIE
   QUELLES SONT LES ORIENTATIONS INTERDITES *)

(* ---------------------------------- *)
FUNCTION APEUPRES(R1,R2:REAL):BOOLEAN;
(* DETERMINE SI 2 NOMBRES SONT +OU- PROCHES EN VALEUR ABSOLUE.
   '+ OU -' DEPEND DE 'FPP',FIXE ACTUELLEMENT A 0.8 *)

CONST FPP=0.8; (* FACTEUR A PEU PRES *)
BEGIN
  IF (ABS(R2)*FPP<=ABS(R1))
  AND (ABS(R2)/FPP>=ABS(R1))
    THEN APEUPRES:=TRUE
    ELSE APEUPRES:=FALSE
END;
(* ------------------------------- *)
PROCEDURE TESTOR1;
(* TEST D'ORientation nro 1 *)

BEGIN
  IF B*C<0
  THEN IF APEUPRES(2/3,B/C)
       THEN BEGIN
              MULX:=-2/3;MULY:=-1/3;
              ORIENTATION[NRO,1]:=FALSE;
              ORIENTATION[NRO,2]:=FALSE;
              ORIENTATION[NRO,5]:=FALSE;
              ORIENTATION[NRO,6]:=FALSE;
            END
       ELSE IF APEUPRES(1/3,B/C)
            THEN BEGIN
                   ORIENTATION[NRO,3]:=FALSE;
                   ORIENTATION[NRO,4]:=FALSE;
                 END
END;
  (* ------------------------------- *)
PROCEDURE TESTOR2;
(* TEST D'ORientation nro 2 *)

BEGIN
  IF A*C<0
  THEN BEGIN
         IF APEUPRES(A,C)
          THEN BEGIN
                 ORIENTATION[NRO,1]:=FALSE;
                 MULX:=2/3;MULY:=-2/3;
               END;
         IF APEUPRES(A/2,C)
          THEN ORIENTATION[NRO,5]:=FALSE;
         IF APEUPRES(A,C/2)
          THEN ORIENTATION[NRO,3]:=FALSE;
       END
  ELSE BEGIN
         IF APEUPRES(A,C)
```

```
          THEN ORIENTATION[NRO,2]:=FALSE;
        IF APEUPRES(A/2,C)
          THEN ORIENTATION[NRO,6]:=FALSE;
        IF APEUPRES(A,C/2)
          THEN ORIENTATION[NRO,4]:=FALSE;
      END
END;
(* ------------------------------ *)
PROCEDURE TESTOR3;
(* TEST D'ORientation nro 3 *)

BEGIN
  IF A*B<0
  THEN IF APEUPRES(2/3,B/A)
       THEN BEGIN
              MULX:=2/3;MULY:=-2/3;
              ORIENTATION[NRO,1]:=FALSE;
              ORIENTATION[NRO,3]:=FALSE;
            END
       ELSE IF APEUPRES(1/3,B/A)
            THEN ORIENTATION[NRO,5]:=FALSE
  ELSE IF APEUPRES(2/3,B/A)
       THEN BEGIN
              ORIENTATION[NRO,2]:=FALSE;
              ORIENTATION[NRO,4]:=FALSE;
            END
       ELSE IF APEUPRES(1/3,B/A)
            THEN ORIENTATION[NRO,6]:=FALSE;
END;
(* ------------------------------- *)
PROCEDURE TESTOR4;
(* TEST D'ORientation nro 4;separe en 2 parties pcq trop long  *)

  PROCEDURE FIRSTPART;
  BEGIN
  IF A*C<0
       THEN BEGIN
              IF APEUPRES(A/C,(2*A)/((3*B)+(2*A)))
              THEN BEGIN
                     ORIENTATION[NRO,1]:=FALSE;
                     MULX:=2/3;MULY:=-2/3
                   END;
              IF APEUPRES(A/C,A/((3*B)+(2*A)))
              THEN ORIENTATION[NRO,3]:=FALSE;
              IF APEUPRES(A/C,(2*A)/((3*B)+A))
              THEN ORIENTATION[NRO,5]:=FALSE;
              IF ABS(2*A)<ABS(3*B)
              THEN BEGIN
                     IF APEUPRES(A/C,(2*A)/((3*B)-(2*A)))
                     THEN ORIENTATION[NRO,2]:=FALSE;
                     IF APEUPRES(A/C,A/((3*B)-(2*A)))
                     THEN ORIENTATION[NRO,4]:=FALSE;
                   END;
              IF ABS(A)<ABS(3*B)
              THEN IF APEUPRES(A/C,(2*A)/((3*B)-A))
                   THEN ORIENTATION[NRO,6]:=FALSE;
            END
       ELSE BEGIN
              IF ABS(2*A)>ABS(3*B)
              THEN BEGIN
                     IF APEUPRES(A/C,(2*A)/((3*B)-(2*A)))
                     THEN ORIENTATION[NRO,2]:=FALSE;
                     IF APEUPRES(A/C,A/((3*B)-(2*A)))
                     THEN ORIENTATION[NRO,6]:=FALSE;
                   END;
```

```
                IF ABS(A)>ABS(3*B)
                THEN IF APEUPRES(A/C,(2*A)/((3*B)-A))
                     THEN ORIENTATION[NRO,4]:=FALSE;
             END
    END;
   (* -------------------------- *)
   PROCEDURE SECONDPART;
   BEGIN
       IF A*C>0
        THEN BEGIN
                IF APEUPRES(A/C,(2*A)/((3*B)-(2*A)))
                THEN BEGIN
                       ORIENTATION[NRO,2]:=FALSE;
                       MULX:=-2/3;MULY:=-2/3
                     END;
                IF APEUPRES(A/C,A/((3*B)-(2*A)))
                THEN ORIENTATION[NRO,4]:=FALSE;
                IF APEUPRES(A/C,(2*A)/((3*B)-A))
                THEN ORIENTATION[NRO,6]:=FALSE;
                IF ABS(2*A)<ABS(3*B)
                THEN BEGIN
                       IF APEUPRES(A/C,(2*A)/((3*B)+(2*A)))
                       THEN ORIENTATION[NRO,1]:=FALSE;
                       IF APEUPRES(A/C,A/((3*B)+(2*A)))
                       THEN ORIENTATION[NRO,3]:=FALSE;
                     END;
                IF ABS(A)<ABS(3*B)
                THEN IF APEUPRES(A/C,(2*A)/((3*B)+A))
                     THEN ORIENTATION[NRO,5]:=FALSE;
             END
         ELSE BEGIN
                IF ABS(2*A)>ABS(3*B)
                THEN BEGIN
                       IF APEUPRES(A/C,(2*A)/((3*B)+(2*A)))
                       THEN ORIENTATION[NRO,1]:=FALSE;
                       IF APEUPRES(A/C,A/((3*B)+(2*A)))
                       THEN ORIENTATION[NRO,3]:=FALSE;
                     END;
                IF ABS(A)>ABS(3*B)
                THEN IF APEUPRES(A/C,(2*A)/((3*B)+A))
                     THEN ORIENTATION[NRO,5]:=FALSE;
             END
      END;
     (* -------------------------- *)

BEGIN
   IF A*B>0
   THEN FIRSTPART
   ELSE SECONDPART
 END;
(* ---------------------------------- *)


(* les differentes procedures PLi determinent les sommets,font appel aux
   tests d'orientation,memorisent le type du plan et garnissent le tableau
   pour les axes negatifs *)


PROCEDURE PL1;
BEGIN
   TYPEDUPLAN[NRO]:=1;
   NBSOMMET[NRO]:=0;
   MULX:=-2/3;
   MULY:=-2/3
```

```
END;

PROCEDURE PL2;
BEGIN
  TYPEDUPLAN[NRO]:=2;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=0;PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=1;PLAN3[NRO,2].P2:=0;PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=1;PLAN3[NRO,3].P2:=1;PLAN3[NRO,3].P3:=0;
  PLAN3[NRO,4].P1:=0;PLAN3[NRO,4].P2:=1;PLAN3[NRO,4].P3:=0;
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL3;
BEGIN
  TYPEDUPLAN[NRO]:=3;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=1;PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=0;PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=0;PLAN3[NRO,3].P3:=1;
  PLAN3[NRO,4].P1:=1;PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=1;
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL4;
BEGIN
  TYPEDUPLAN[NRO]:=4;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=ABS(B/C);PLAN3[NRO,1].P2:=1;PLAN3[NRO,1].P3:=(-B/C);
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=1;PLAN3[NRO,2].P3:=(-B/C);
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=-1;PLAN3[NRO,3].P3:=(B/C);
  PLAN3[NRO,4].P1:=ABS(B/C);PLAN3[NRO,4].P2:=-1;PLAN3[NRO,4].P3:=(B/C);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR1;
END;

PROCEDURE PL5;
BEGIN
  TYPEDUPLAN[NRO]:=5;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=0;PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=1;PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=1;PLAN3[NRO,3].P3:=1;
  PLAN3[NRO,4].P1:=0;PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=1;
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL6;
BEGIN
  TYPEDUPLAN[NRO]:=6;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=1;PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=(-A/C);
  PLAN3[NRO,2].P1:=1;PLAN3[NRO,2].P2:=ABS(A/C);PLAN3[NRO,2].P3:=(-A/C);
  PLAN3[NRO,3].P1:=-1;PLAN3[NRO,3].P2:=ABS(A/C);PLAN3[NRO,3].P3:=(A/C);
  PLAN3[NRO,4].P1:=-1;PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=(A/C);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR2;
END;

PROCEDURE PL7;
BEGIN
  TYPEDUPLAN[NRO]:=7;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=1;PLAN3[NRO,1].P2:=(-A/B);PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=-1;PLAN3[NRO,2].P2:=(A/B);PLAN3[NRO,2].P3:=0;
```

```
  PLAN3[NRO,3].P1:=-1;PLAN3[NRO,3].P2:=(A/B);PLAN3[NRO,3].P3:=ABS(A/B);
  PLAN3[NRO,4].P1:=1;PLAN3[NRO,4].P2:=(-A/B);PLAN3[NRO,4].P3:=ABS(A/B);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR3;
END;

PROCEDURE PL8;
BEGIN
  TYPEDUPLAN[NRO]:=8;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=1;PLAN3[NRO,1].P2:=(-A/B);PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=-1;PLAN3[NRO,2].P2:=(A/B);PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=-2;PLAN3[NRO,3].P2:=(A/B);PLAN3[NRO,3].P3:=(A/C);
  PLAN3[NRO,4].P1:=0;PLAN3[NRO,4].P2:=-(A/B);PLAN3[NRO,4].P3:=(A/C);
  IF ((A*C)<0)
   THEN
    BEGIN
      NBSOMMET[NRO]:=6;
      PLAN3[NRO,5].P1:=0;PLAN3[NRO,5].P2:=(A/B);PLAN3[NRO,5].P3:=(-A/C);
      PLAN3[NRO,6].P1:=2;PLAN3[NRO,6].P2:=(-A/B);PLAN3[NRO,6].P3:=(-A/C);
    END;
  MULX:=-2/3;MULY:=-2/3;
  TESTOR4;
END;

PROCEDURE PL9;
BEGIN
  TYPEDUPLAN[NRO]:=9;
  NBSOMMET[NRO]:=0;
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL10;
BEGIN
  TYPEDUPLAN[NRO]:=10;
  IF (D/C)<0 THEN AX3NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=ABS(2*D/C);PLAN3[NRO,1].P2:=ABS(2*D/C);PLAN3[NRO,1].P3
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=ABS(2*D/C);PLAN3[NRO,2].P3:=(D/C);
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=0;PLAN3[NRO,3].P3:=(D/C);
  PLAN3[NRO,4].P1:=ABS(2*D/C);PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=(D/C);
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL11;
BEGIN
  TYPEDUPLAN[NRO]:=11;
  IF (D/B)<0 THEN AX2NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=ABS(2*D/B);PLAN3[NRO,1].P2:=(D/B);PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=(D/B);PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=(D/B);PLAN3[NRO,3].P3:=ABS(2*D/B);
  PLAN3[NRO,4].P1:=ABS(2*D/B);PLAN3[NRO,4].P2:=(D/B);
  PLAN3[NRO,4].P3:=ABS(2*D/B);
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL12;
BEGIN
  TYPEDUPLAN[NRO]:=12;
  IF (D/B)<0 THEN AX2NEG[NRO]:=TRUE;
  IF (D/C)<0 THEN AX3NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=ABS(2*D/B);PLAN3[NRO,1].P2:=(D/B);PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=(D/B);PLAN3[NRO,2].P3:=0;
```

```
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=0;PLAN3[NRO,3].P3:=(D/C);
  PLAN3[NRO,4].P1:=ABS(2*D/B);PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=(D/C);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR1;
END;

PROCEDURE PL13;
BEGIN
  TYPEDUPLAN[NRO]:=13;
  IF (D/A)<0 THEN AX1NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=(D/A);PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=(D/A);PLAN3[NRO,2].P2:=ABS(2*D/A);PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=(D/A);PLAN3[NRO,3].P2:=ABS(2*D/A);PLAN3[NRO,3].P3:=ABS(2
  PLAN3[NRO,4].P1:=(D/A);PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=ABS(2*D/A);
  MULX:=(-2/3);MULY:=(-2/3);
END;

PROCEDURE PL14;
BEGIN
  TYPEDUPLAN[NRO]:=14;
  IF (D/A)<0 THEN AX1NEG[NRO]:=TRUE;
  IF (D/C)<0 THEN AX3NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=(D/A);PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=(D/A);PLAN3[NRO,2].P2:=ABS(2*D/A);PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=ABS(2*D/A);PLAN3[NRO,3].P3:=(D/C);
  PLAN3[NRO,4].P1:=0;PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=(D/C);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR2;
END;

PROCEDURE PL15;
BEGIN
  TYPEDUPLAN[NRO]:=15;
  IF (D/A)<0 THEN AX1NEG[NRO]:=TRUE;
  IF (D/B)<0 THEN AX2NEG[NRO]:=TRUE;
  NBSOMMET[NRO]:=4;
  PLAN3[NRO,1].P1:=(D/A);PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=(D/B);PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=0;PLAN3[NRO,3].P2:=(D/B);PLAN3[NRO,3].P3:=ABS(2*D/A);
  PLAN3[NRO,4].P1:=(D/A);PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=ABS(2*D/A);
  MULX:=-2/3;MULY:=-2/3;
  TESTOR3;
END;

PROCEDURE PL16;
VAR I1,I2,I3:REAL;
BEGIN
  TYPEDUPLAN[NRO]:=16;
  I1:=D/A;I2:=D/B;I3:=D/C;
  IF I1<0 THEN AX1NEG[NRO]:=TRUE;
  IF I2<0 THEN AX2NEG[NRO]:=TRUE;
  IF I3<0 THEN AX3NEG[NRO]:=TRUE;
  PLAN3[NRO,1].P1:=I1;PLAN3[NRO,1].P2:=0;PLAN3[NRO,1].P3:=0;
  PLAN3[NRO,2].P1:=0;PLAN3[NRO,2].P2:=I2;PLAN3[NRO,2].P3:=0;
  PLAN3[NRO,3].P1:=-I1;PLAN3[NRO,3].P2:=I2;PLAN3[NRO,3].P3:=I3;
  PLAN3[NRO,4].P1:=0;PLAN3[NRO,4].P2:=0;PLAN3[NRO,4].P3:=I3;
  IF I3>0
  THEN NBSOMMET[NRO]:=4
  ELSE
   BEGIN
     NBSOMMET[NRO]:=6;
     PLAN3[NRO,5].P1:=I1;
     PLAN3[NRO,5].P2:=I2;
```

```
      PLAN3[NRO,5].P3:=-I3;
      PLAN3[NRO,6].P1:=(2*I1);
      PLAN3[NRO,6].P2:=0;
      PLAN3[NRO,6].P3:=-I3;
    END;
   MULY:=-2/3;
   IF A*B>0 THEN MULX:=-2/3
            ELSE MULX:=2/3;
   TESTOR4;
END;
(* ------------------------------- *)


(* debut de  CHOIXPLAN  *)

BEGIN
 IF D=0
 THEN
  IF A=0
  THEN
   IF B=0
   THEN
    IF C=0
     THEN PL1 (*PLAN INDETERMINE*)
     ELSE PL2 (*CZ=0 -> PLAN HORIZONTAL*)
   ELSE
    IF C=0
     THEN PL3 (*BY=0 -> PLAN DE FACE*)
     ELSE PL4 (*BY+CZ=0*)
  ELSE
   IF B=0
   THEN
    IF C=0
     THEN PL5 (*AX=0 ->PLAN OBLIQUE*)
     ELSE PL6 (*AX+CZ=0*)
   ELSE
    IF C=0
    THEN PL7 (*AX+BY=0*)
    ELSE PL8 (*AX+BY+CZ=0*)
 ELSE
  IF A=0
  THEN
   IF B=0
   THEN
    IF C=0
    THEN PL9 (*0=D ->IMPOSSIBLE*)
    ELSE PL10 (*CZ=D ->//HORIZONTAL*)
   ELSE
    IF C=0
    THEN PL11 (*BY=D ->//FACE*)
    ELSE PL12 (*BY+CZ=D*)
  ELSE
   IF B=0
   THEN
    IF C=0
    THEN PL13 (*AX=D ->//OBLIQUE*)
    ELSE PL14 (*AX+CZ=D*)
   ELSE
    IF C=0
    THEN PL15 (*AX+BY=D*)
    ELSE PL16 (*AX+BY+CZ=D*)
END;

(* ------------------------------- *)
BEGIN
END,
```

```
(*$S+*)
UNIT INTER33;INTRINSIC CODE 14;
INTERFACE
(*======*)
USES DECLAR33;

PROCEDURE MESSAGE(CHAINE:STRING);
FUNCTION INTERVALLE(R1,INT1,INT2:REAL):BOOLEAN;
PROCEDURE PROLONGER(NROPLAN:INTEGER;INTER:POINT;DIR:SENS);
PROCEDURE INTERPLAN(N1,N2:INTEGER);
(* ---------------------------------*)
IMPLEMENTATION
(*==========*)
PROCEDURE MESSAGE;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(0,10);
  WRITELN('************************************');
  WRITE('* ',CHAINE);
  GOTOXY(34,11);WRITELN('*');
WRITELN('************************************');
END;
(* ------------------------------ *)
function intervalle;
(*determine si un nombre donne (r1) est compris dans l'intervalle [int1,int

begin
  if (((r1<=int1) and (r1>=int2))
   or ((r1>=int1) and (r1<=int2)))
     then intervalle:=true
     else intervalle:=false
end;
(* ------------------------------- *)
procedure prolonger;
(* prolonge le plan NROPLAN vers INTER dans le sens DIR *)

var a,b,c,d:real;

   procedure anul;
   begin
     if intervalle(inter.p1,plan3[nroplan,1].p1,
                        plan3[nroplan,2].p1)
      then
      else
       if inter.p1*plan3[nroplan,1].p1>0
        then begin
               plan3[nroplan,1].p1:=inter.p1;
               plan3[nroplan,4].p1:=inter.p1
             end
        else begin
               plan3[nroplan,2].p1:=inter.p1;
               plan3[nroplan,3].p1:=inter.p1
             end;
     if intervalle(inter.p2,plan3[nroplan,1].p2,
                        plan3[nroplan,2].p2)
      then
      else if b=0
       then if inter.p2*plan3[nroplan,2].p2>0
        then begin
               plan3[nroplan,1].p2:=inter.p2;
               plan3[nroplan,2].p2:=inter.p2
             end
        else begin
```

```
            plan3[nroplan,3].p2:=inter.p2;
            plan3[nroplan,4].p2:=inter.p2
          end;
end;

procedure anonnul;
var decalage:real;
begin
  if b<>0
   then if intervalle(inter.p1,plan3[nroplan,1].p1,
                      plan3[nroplan,2].p1)
    then
    else
     if inter.p1*plan3[nroplan,1].p1>0
      then begin
         decalage:=plan3[nroplan,1].p1-plan3[nroplan,4].p1;
         plan3[nroplan,1].p1:=inter.p1;
         plan3[nroplan,1].p2:=(d-(a*inter.p1)
                              -(c*plan3[nroplan,1].p3))/b;
         plan3[nroplan,4].p1:=inter.p1-decalage;
         plan3[nroplan,4].p2:=(d-(a*plan3[nroplan,4].p1)
                              -(c*plan3[nroplan,4].p3))/b;
          end
      else begin
          decalage:=plan3[nroplan,1].p1-plan3[nroplan,4].p1;
          plan3[nroplan,2].p1:=inter.p1;
          plan3[nroplan,2].p2:=(d-(a*inter.p1)
                               -(c*plan3[nroplan,2].p3))/b;
          plan3[nroplan,3].p1:=inter.p1-decalage;
          plan3[nroplan,3].p2:=(d-(a*plan3[nroplan,3].p1)
                               -(c*plan3[nroplan,3].p3))/b;
          end
    else if intervalle(inter.p2,plan3[nroplan,1].p2,
                               plan3[nroplan,2].p2)
          then
          else
           if inter.p2*plan3[nroplan,2].p2>0
           then begin
            plan3[nroplan,3].p2:=inter.p2;
            plan3[nroplan,2].p2:=inter.p2;
                end
           else begin
            plan3[nroplan,1].p2:=inter.p2;
            plan3[nroplan,4].p2:=inter.p2;
                end
end;

procedure prolhor;
begin
  if a=0
   then anul
   else anonnul
end;

procedure prolvert;
begin
  if c=0
   then if intervalle(inter.p3,plan3[nroplan,1].p3,
                      plan3[nroplan,4].p3)
    then
    else if inter.p3*plan3[nroplan,4].p3>0
      then begin
             plan3[nroplan,3].p3:=inter.p3;
             plan3[nroplan,4].p3:=inter.p3
           end
```

```
          else begin
                 plan3[nroplan,1].p3:=inter.p3;
                 plan3[nroplan,2].p3:=inter.p3
               end
       else if a<>0
            then if intervalle(inter.p3,plan3[nroplan,1].p3,plan3[nroplan,
             then
             else if inter.p3*plan3[nroplan,4].p3>0
              then begin
                  plan3[nroplan,4].p3:=inter.p3;
                  plan3[nroplan,4].p1:=(d-(b*plan3[nroplan,4].p2)
                                        -(c*inter.p3))/a;
                  plan3[nroplan,3].p3:=inter.p3;
                  plan3[nroplan,3].p1:=(d-(b*plan3[nroplan,3].p2)
                                        -(c*inter.p3))/a;
                   end
              else begin
                  plan3[nroplan,1].p3:=inter.p3;
                  plan3[nroplan,1].p1:=(d-(b*plan3[nroplan,1].p2)
                                        -(c*inter.p3))/a;
                  plan3[nroplan,2].p3:=inter.p3;
                  plan3[nroplan,2].p1:=(d-(b*plan3[nroplan,2].p2)
                                        -(c*inter.p3))/a;
                   end
            else if b<>0
                  then if intervalle(inter.p3,plan3[nroplan,1].p3,
                                     plan3[nroplan,4].p3)
                  then
                  else if inter. #"  !"##""#  !" #!" #" ! !  !" "!#!"! "
                   plan3[nroplan,2].p3:=inter.p3;
                   plan3[nroplan,2].p2:=(d-(c*inter.p3))/b;
                      end
end;


begin
  a:=aa[nroplan];
  b:=bb[nroplan];
  c:=cc[nroplan];
  d:=dd[nroplan];
  if dir=horizontal
   then prolhor (* prolonger dans le plan horizontal  *)
   else prolvert;(* vertical *)
end;
(* ------------------------------- *)

PROCEDURE INTERPLAN;
(* ETUDIE LA RELATION ENTRE 2 PLANS (N1 ET N2) ET S'ILS SONT SECANTS
   DETERMINE DES PORTIONS DE PLANS PERMETTANT DE VOIR L'INTERSECTION *)

VAR A1,A2,B1,B2,C1,C2,D1,D2 : REAL;
    AB,AC,AD,BC,BD,CD:REAL;

  function max(r1,r2:real):real;
  begin
    if abs(r1)>=abs(r2)
     then max:=r1
     else max:=r2
  end;
  (* ------------------------------- *)
  PROCEDURE NIX1NIX2;
  BEGIN
    CD:=(CC[N2]*DD[N1])-(CC[N1]*DD[N2]);
    IF CD=0
     THEN MESSAGE('PLANS CONFONDUS')
```

```
      ELSE MESSAGE('PLANS PARALLELES');
  END;
   (* ------------------------- *)
  PROCEDURE NIX1NIX3;
  BEGIN
    BD:=(BB[N2]*DD[N1])-(BB[N1]*DD[N2]);
    IF BD=0
     THEN MESSAGE('PLANS CONFONDUS')
     ELSE MESSAGE('PLANS PARALLELES');
  END;
   (* ------------------------- *)
PROCEDURE NIX2NIX3;
  BEGIN
    AD:=(AA[N2]*DD[N1])-(AA[N1]*DD[N2]);
    IF AD=0
     THEN MESSAGE('PLANS CONFONDUS')
     ELSE MESSAGE('PLANS PARALLELES');
  END;
   (* ------------------------- *)
PROCEDURE PASDEX1;

  BEGIN
    IF BB[N1]<>0
     THEN BEGIN
            B1:=BB[N1];C1:=CC[N1];D1:=DD[N1];
            B2:=BB[N2];C2:=CC[N2];D2:=DD[N2]
          END
     ELSE BEGIN
            B1:=BB[N2];C1:=CC[N2];D1:=DD[N2];
            B2:=BB[N1];C2:=CC[N1];D2:=DD[N1]
          END;
    BC:=(B2*C1)-(B1*C2);BD:=(B2*D1)-(B1*D2);
    IF BC=0
     THEN
      IF BD=0
       THEN MESSAGE('PLANS CONFONDUS')
       ELSE MESSAGE('PLANS PARALLELES')
     ELSE
      BEGIN
        INTER1.P3:=BD/BC;
        INTER1.P2:=(D1-(C1*INTER1.P3))/B1;
        INTER1.P1:=0;
        PROLONGER(N1,INTER1,VERTICAL);
        PROLONGER(N2,INTER1,VERTICAL);
        INTER2.P1:=MAX(PLAN3[N1,1].P1,
                       PLAN3[N2,1].P1);
        INTER2.P2:=INTER1.P2;
        INTER2.P3:=INTER1.P3;
        PROLONGER(N1,INTER2,HORIZONTAL);
        PROLONGER(N2,INTER2,HORIZONTAL);
      end
  END;
   (* ------------------------- *)
PROCEDURE PASDEX2;

  BEGIN
    IF AA[N1]<>0
     THEN BEGIN
            A1:=AA[N1];C1:=CC[N1];D1:=DD[N1];
            A2:=AA[N2];C2:=CC[N2];D2:=DD[N2]
          END
     ELSE BEGIN
            A1:=AA[N2];C1:=CC[N2];D1:=DD[N2];
            A2:=AA[N1];C2:=CC[N1];D2:=DD[N1]
          END;
```

```
    AC:=(A2*C1)-(A1*C2);AD:=(A2*D1)-(A1*D2);
    IF AC=0
     THEN
      IF AD=0
       THEN MESSAGE('PLANS CONFONDUS')
       ELSE MESSAGE('PLANS PARALLELES')
     ELSE
      BEGIN
        INTER1.P3:=AD/AC;
        INTER1.P1:=(D1-(C1*INTER1.P3))/A1;
        INTER1.P2:=0;
        PROLONGER(N1,INTER1,VERTICAL);
        PROLONGER(N2,INTER1,VERTICAL);
        INTER2.P2:=MAX(PLAN3[N1,2].P2,
                       PLAN3[N2,2].P2);
        INTER2.p1:=INTER1.P1;
        INTER2.P3:=INTER1.P3;
        PROLONGER(N1,INTER2,HORIZONTAL);
        PROLONGER(N2,INTER2,HORIZONTAL);
      END
  END;
   (* ---------------------------- *)
PROCEDURE PASDEX3;

  BEGIN
    IF AA[N1]<>0
     THEN BEGIN
            A1:=AA[N1];B1:=BB[N1];D1:=DD[N1];
            A2:=AA[N2];B2:=BB[N2];D2:=DD[N2]
          END
     ELSE BEGIN
            A1:=AA[N2];B1:=BB[N2];D1:=DD[N2];
            A2:=AA[N1];B2:=BB[N1];D2:=DD[N1]
          END;
    AB:=(A2*B1)-(A1*B2);AD:=(A2*D1)-(A1*D2);
    IF AB=0
     THEN
      IF aD=0
       THEN MESSAGE('PLANS CONFONDUS')
       ELSE MESSAGE('PLANS PARALLELES')
     ELSE
      BEGIN
        INTER1.P2:=AD/AB;
        INTER1.P1:=(D1-(b1*INTER1.P2))/A1;
        INTER1.P3:=0;
        PROLONGER(N1,INTER1,HORIZONTAL);
        PROLONGER(N2,INTER1,HORIZONTAL);
        INTER2.P3:=MAX(PLAN3[N1,4].P3,
                       PLAN3[N2,4].P3);
        INTER2.P2:=INTER1.P2;
        INTER2.P1:=INTER1.P1;
        PROLONGER(N1,INTER2,VERTICAL);
        PROLONGER(N2,INTER2,VERTICAL);
      END
  END;
   (* ---------------------------- *)

  PROCEDURE CASGENERAL;
  (* CETTE PROCEDURE EST FORT DECOUPEE:

     IF AB=0
     THEN ABNUL
     ELSE ( ABNONNUL );
            IF AC=0
            THEN ACNUL
```

```
          ELSE ( ACNONNUL );
                .........
                IF AD=0
                THEN ADNUL
                ELSE ADNONNUL;
*)

VAR NUMERO:INTEGER;
    TAMPON:POINT;

   procedure abnul;
   begin
   if ac=0
   then
    if ad=0
    then message('les 2 plans sont confondus')
    else message('les 2 plans sont paralleles')
   else
    if ad=0
    then if d1<>0
           then begin
                  inter1.p1:=d1/a1;
                  inter1.p2:=0;
                  inter1.p3:=0;
                  prolonger(n2,inter1,horizontal);
                  inter2.p1:=0;
                  inter2.p2:=d1/b1;
                  inter2.p3:=0;
                  prolonger(n2,inter2,horizontal);
                end
           else begin
                  inter1.p1:=1;
                  inter1.p2:=-a1/b1;
                  inter1.p3:=0;
                  prolonger(n2,inter1,horizontal);
                  inter2.p1:=-1;
                  inter2.p2:=a1/b1;
                  inter2.p3:=0;
                  prolonger(n2,inter2,horizontal);
                end
    else begin
           inter1.p3:=ad/ac;
           inter1.p2:=0;
           inter1.p1:=(d1-(c1*inter1.p3))/a1;
           prolonger(n1,inter1,vertical);
           if (a1=0) and (a2=0)
            then prolonger(n2,inter1,horizontal)
            else prolonger(n2,inter1,vertical);
           inter2.p2:=max(plan3[n1,2].p2,
                          plan3[n2,2].p2);
           inter2.p3:=inter1.p3;
           inter2.p1:=(d1-(c1*inter2.p3)-(b1*inter2.p2))/a1;
           if inter2.p2=plan3[n1,2].p2
           then
            begin
              tampon.p3:=0;
              tampon.p2:=inter2.p2;
              if a2<>0
               then tampon.p1:=(d2-(b2*inter2.p2))/a2
               else tampon.p1:=inter2.p1;
              prolonger(n2,tampon,horizontal);
              prolonger(n1,inter2,vertical);
              if (a2=0) and (b2=0)
               then prolonger(n2,inter2,horizontal)
               else prolonger(n2,inter2,vertical);
```

```
              end
            else
             begin
               tampon.p3:=0;
               tampon.p2:=inter2.p2;
               tampon.p1:=(d1-(b1*inter2.p2))/a1;
               prolonger(n1,tampon,horizontal);
               prolonger(n1,inter2,vertical);
               if (a2=0) and (b2=0)
                then prolonger(n2,inter2,horizontal)
                else prolonger(n2,inter2,vertical);
             end
          end
    end;

    procedure acnul;
    begin
     if ad=0
     then begin
            inter1.p1:=d1/a1;
            inter1.p2:=0;
            inter1.p3:=0;
            prolonger(n2,inter1,horizontal);
            if d1<>0
             then begin
                    inter2.p1:=0;
                    inter2.p2:=0;
                    inter2.p3:=d1/c1;
                    prolonger(n2,inter2,vertical);
                  end
             else begin
                    inter2.p1:=-1;
                    inter2.p2:=0;
                    inter2.p3:=a1/c1;
                    prolonger(n2,inter2,horizontal);
                    prolonger(n2,inter2,vertical);
                  end
          end

      else begin
            inter1.p2:=ad/ab;
            inter1.p3:=0;
            inter1.p1:=(d1-(b1*inter1.p2))/a1;
            prolonger(n1,inter1,horizontal);
            prolonger(n2,inter1,horizontal);
            inter2.p3:=max(plan3[n1,4].p3,
                           plan3[n2,4].p3);
            inter2.p2:=inter1.p2;
            inter2.p1:=(d1-(c1*inter2.p3)-(b1*inter2.p2))/a1;
            if plan3[n1,4].p3=inter2.p3
            then begin
                   prolonger(n1,inter2,horizontal);
                   prolonger(n2,inter2,vertical);
                 end
            else begin
                   prolonger(n1,inter2,vertical);
                   prolonger(n2,inter2,horizontal);
                 end
          end
   end;

procedure CHINT1;(* CHercher INTersection 1 *)
  begin
    tampon.p3:=plan3[n1,4].p3;
    tampon.p2:=(ad-(ac*tampon.p3))/ab;
```

```
  tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
  if not ((intervalle(tampon.p1,plan3[n1,3].p1,plan3[n1,4].p1))
        and (intervalle(tampon.p2,plan3[n1,3].p2,plan3[n1,4].p2)))
   then
   begin
     tampon.p3:=plan3[n2,4].p3;
     tampon.p2:=(ad-(ac*tampon.p3))/ab;
     tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
     if not ((intervalle(tampon.p1,plan3[n2,3].p1,plan3[n2,4].p1))
           and (intervalle(tampon.p2,plan3[n2,3].p2,plan3[n2,4].p2)))
      then
      begin
        tampon.p2:=plan3[n1,2].p2;
        tampon.p3:=(ad-(ab*tampon.p2))/ac;
        tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
        if not ((intervalle(tampon.p1,plan3[n1,2].p1,plan3[n1,3].p1))
              and (intervalle(tampon.p3,plan3[n1,2].p3,plan3[n1,3].p3))
         then
         begin
           tampon.p2:=plan3[n2,2].p2;
           tampon.p3:=(ad-(ab*tampon.p2))/ac;
           tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
           if not((intervalle(tampon.p1,plan3[n2,2].p1,plan3[n2,3].p1))
                 and (intervalle(tampon.p3,plan3[n2,2].p3,plan3[n2,3].p3
            then
            begin
              tampon.p3:=plan3[n1,1].p3;
              tampon.p2:=(ad-(ac*tampon.p3))/ab;
              tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
              if not((intervalle(tampon.p1,plan3[n1,1].p1,plan3[n1,2].p1
                   and (intervalle(tampon.p2,plan3[n1,1].p2,plan3[n1,2].p
               then
               begin
                 tampon.p3:=plan3[n2,i].p3;
                 tampon.p2:=(ad-(ac*tampon.p3))/ab;
                 tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
               end
            end
         end
      end
   end;
  inter1:=tampon
end;

procedure CHINT2;(* CHercher INTersection 2 *)
begin
  tampon.p3:=plan3[n1,4].p3;
  tampon.p2:=(ad-(ac*tampon.p3))/ab;
  tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
  if not ((intervalle(tampon.p1,plan3[n1,3].p1,plan3[n1,4].p1))
        and (intervalle(tampon.p2,plan3[n1,3].p2,plan3[n1,4].p2)))
   then
   begin
     tampon.p3:=plan3[n2,4].p3;
     tampon.p2:=(ad-(ac*tampon.p3))/ab;
     tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
     if not ((intervalle(tampon.p1,plan3[n2,3].p1,plan3[n2,4].p1))
           and (intervalle(tampon.p2,plan3[n2,3].p2,plan3[n2,4].p2)))
      then
      begin
        tampon.p2:=plan3[n1,2].p2;
        tampon.p3:=(ad-(ab*tampon.p2))/ac;
        tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
        if not ((intervalle(tampon.p1,plan3[n1,2].p1,plan3[n1,3].p1))
              and (intervalle(tampon.p3,plan3[n1,2].p3,plan3[n1,3].p3))
```

```
          then
          begin
            tampon.p2:=plan3[n2,2].p2;
            tampon.p3:=(ad-(ab*tampon.p2))/ac;
            tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
            if not((intervalle(tampon.p1,plan3[n2,2].p1,plan3[n2,3].p1))
                  and (intervalle(tampon.p3,plan3[n2,2].p3,plan3[n2,3].p3
             then
             begin
               tampon.p2:=plan3[n1,1].p2;
               tampon.p3:=(ad-(ab*tampon.p2))/ac;
               tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
               if not((intervalle(tampon.p1,plan3[n1,1].p1,plan3[n1,4].p1
                    and (intervalle(tampon.p3,plan3[n1,1].p3,plan3[n1,4].p
                then
                begin
                  tampon.p2:=plan3[n2,1].p2;
                  tampon.p3:=(ad-(ab*tampon.p2))/ac;
                  tampon.p1:=(d1-(b1*tampon.p2)-(c1*tampon.p3))/a1;
                end
             end
          end
       end
    end;
  inter2:=tampon
end;

procedure adnul;
begin
  inter2.p3:=max(plan3[n1,4].p3,plan3[n2,4].p3);
  inter2.p2:=-(ac*inter2.p3)/ab;
  inter2.p1:=(d1-(b1*inter2.p2)-(c1*inter2.p3))/a1;
  if inter2.p3=plan3[n1,4].p3
   then begin
          tampon.p3:=0;
          tampon.p2:=inter2.p2;
          tampon.p1:=(d1-(b1*inter2.p2))/a1;
          prolonger(n1,tampon,horizontal);
          prolonger(n2,inter2,vertical);
          tampon.p3:=0;
          tampon.p2:=inter2.p2;
          if a2<>0
           then tampon.p1:=(d2-(b2*inter2.p2))/a2
           else tampon.p1:=inter2.p1;
          prolonger(n2,tampon,horizontal);
        end
   else begin
          tampon.p3:=0;
          tampon.p2:=inter2.p2;
          if a2<>0
           then tampon.p1:=(d2-(b2*inter2.p2))/a2
           else tampon.p1:=inter2.p1;
          prolonger(n2,tampon,horizontal);
          prolonger(n1,inter2,vertical);
          tampon.p3:=0;
          tampon.p2:=inter2.p2;
          tampon.p1:=(d1-(b1*inter2.p2))/a1;
          prolonger(n1,tampon,horizontal);
        end
end;

procedure adnonnul;

begin
  inter2.p3:=ad/ac;
```

```
    inter2.p2:=0;
    inter2.p1:=(d1-(c1*inter2.p3))/a1;
    prolonger(n1,inter2,vertical);
    prolonger(n2,inter2,vertical);
    if a2=0
    then
     begin
       tampon.p3:=0;
       tampon.p1:=inter2.p1;
       tampon.p2:=plan3[n2,2].p2;
       prolonger(n2,tampon,horizontal)
     end;
end;

 FUNCTION LIMITE(PT:POINT;NRO:INTEGER):BOOLEAN;
 BEGIN
   LIMITE:=FALSE;
   IF ((INTERVALLE(PT.P3,PLAN3[NRO,1].P3,PLAN3[NRO,2].P3))
   AND (INTERVALLE(PT.P2,PLAN3[NRO,1].P2,PLAN3[NRO,2].P2))
   AND (INTERVALLE(PT.P1,PLAN3[NRO,1].P1,PLAN3[NRO,2].P1)))
   OR ((INTERVALLE(PT.P3,PLAN3[NRO,1].P3,PLAN3[NRO,4].P3))
   AND (INTERVALLE(PT.P2,PLAN3[NRO,1].P2,PLAN3[NRO,4].P2))
   AND (INTERVALLE(PT.P1,PLAN3[NRO,1].P1,PLAN3[NRO,4].P1)))
   OR ((INTERVALLE(PT.P3,PLAN3[NRO,2].P3,PLAN3[NRO,3].P3))
   AND (INTERVALLE(PT.P2,PLAN3[NRO,2].P2,PLAN3[NRO,3].P2))
   AND (INTERVALLE(PT.P1,PLAN3[NRO,2].P1,PLAN3[NRO,3].P1)))
   OR ((INTERVALLE(PT.P3,PLAN3[NRO,3].P3,PLAN3[NRO,4].P3))
   AND (INTERVALLE(PT.P2,PLAN3[NRO,3].P2,PLAN3[NRO,4].P2))
   AND (INTERVALLE(PT.P1,PLAN3[NRO,3].P1,PLAN3[NRO,4].P1)))
    THEN LIMITE:=TRUE
 END;

 procedure acnonnul;
 begin
   inter1.p3:=0;
   inter1.p2:=ad/ab;
   inter1.p1:=(d1-(b1*inter1.p2))/a1;
   prolonger(n1,inter1,horizontal);
   prolonger(n2,inter1,horizontal);
   if ad=0
   then adnul
   else adnonnul;
   if (limite(inter1,n1)) or (limite(inter1,n2))
    then if (limite(inter2,n1)) or (limite(inter2,n2))
          then
          else chint2
    else chint1
 end;

 procedure abnonnul;
 begin
  if ac=0 then acnul
          else acnonnul
 end;

 begin
   if aa[n1]=0
    then begin
           numero:=n1;
           n1:=n2;
           n2:=numero;
         end;
    a1:=aa[n1];b1:=bb[n1];c1:=cc[n1];d1:=dd[n1];
    a2:=aa[n2];b2:=bb[n2];c2:=cc[n2];d2:=dd[n2];
    ab:=(a2*b1)-(a1*b2);ac:=(a2*c1)-(a1*c2);ad:=(a2*d1)-(a1*d2);
```

```
    if ab=0 then abnul
            else abnonnul
  end;
  (* ----------------------------*)
begin
  if typeduplan[n1]=9
  then message('plan n1 impossible')
  else
   if typeduplan[n2]=9
   then message('plan n2 impossible')
   else
    if typeduplan[n1]=1
    then
     if typeduplan[n2]=1
     then message('doublement indetermine')
     else message('plan n1 indetermine')
    else
     if typeduplan[n2]=1
     then message('plan n2 indetermine')
     else
      if (aa[n1]=0) and (aa[n2]=0)
      then
       if (bb[n1]=0) and (bb[n2]=0)
       then nix1nix2
       else
        if (cc[n1]=0) and (cc[n2]=0)
        then nix1nix3
        else pasdex1
      else
       if (bb[n1]=0) and (bb[n2]=0)
       then
        if (cc[n1]=0) and (cc[n2]=0)
        then nix2nix3
        else pasdex2
       else
        if (cc[n1]=0) and (cc[n2]=0)
         then pasdex3
         else casgeneral
end;
(* ------------------------------ *)
BEGIN
END.
```

```
(*$S+,N*)
UNIT DESSIN33;INTRINSIC CODE 15;
INTERFACE
(*======*)
USES APPLESTUFF,UTILUNIT,TURTLEGRAPHICS,
     DECLAR33,CHOIXPLAN,INTER33;

PROCEDURE INITORI;
PROCEDURE UNPLAN(A1,B1,C1,D1:REAL;
                 NRO:INTEGER);
PROCEDURE DEUXPLANS(NRO1,NRO2:INTEGER;
                    A1,B1,C1,D1,
                    A2,B2,C2,D2:REAL);
PROCEDURE SYST33(MATF:MATRICE;
                 MATR:MATREEL);
(* ---------------------------------- *)
IMPLEMENTATION
(*==========*)
PROCEDURE PASSERA2DIM(D3:POINT;VAR D2:COORDONNEE);
(*transforme un point en 3 dimensions "logiques" en 2 dimensions "physiqu

VAR TAMPON:REAL;
BEGIN
  TAMPON:=CX+UNITE*(D3.P1+(D3.P2*MULX));
  D2.XX:=ROUND(TAMPON);
  TAMPON:=CY+UNITE*(D3.P3+(D3.P2*MULY));
  D2.YY:=ROUND(TAMPON);
END;
(* ---------------------------------- *)
PROCEDURE CRAYON(NROCOULEUR:INTEGER;VAR COULEUR:SCREENCOLOR);
(* permet de ne devoir passer qu'un chiffre au lieu du nom de la couleur

BEGIN
  CASE NROCOULEUR OF
   0:COULEUR:=NONE;
   1:COULEUR:=WHITE;
   2:COULEUR:=BLACK;
   3:COULEUR:=REVERSE;
   4:COULEUR:=RADAR;
   5:COULEUR:=BLACK1;
   6:COULEUR:=GREEN;
   7:COULEUR:=VIOLET;
   8:COULEUR:=WHITE1;
   9:COULEUR:=BLACK2;
   10:COULEUR:=ORANGE;
   11:COULEUR:=BLUE;
   12:COULEUR:=WHITE2;
  END;
END;
(* ---------------------------------- *)
PROCEDURE TRDROITE(D21,D22:COORDONNEES;
                   NROCOULEUR,POINTILLE:INTEGER);
(* joint les points D21 et D22 en la couleur correspondant a NROCOULEUR.
   POINTILLE indique si le trait doit etre plein ou pointille (nb:pas en
   implemente *)

VAR COULEUR:SCREENCOLOR;
BEGIN
  (*$R TURTLEGRAPHICS *)
  CRAYON(NROCOULEUR,COULEUR);
  PENCOLOR(NONE);
  MOVETO(D21.XX,D21.YY);
  PENCOLOR(COULEUR);
  MOVETO(D22.XX,D22.YY)
END;
```

```
(* ---------------------------------- *)
PROCEDURE TRAXES;
(* TRacer les AXES *)

VAR NBPOINTS:REAL;
    COORD1,COORD2:COORDONNEE;
BEGIN
  (*$R TURTLEGRAPHICS *)
  COORD1.YY:=CY;COORD2.YY:=CY;
  COORD2.XX:=XMAX;
  IF AXE1NEG
   THEN COORD1.XX:=0
   ELSE COORD1.XX:=CX;
  TRDROITE(COORD1,COORD2,1,0);
  COORD1.XX:=CX;COORD2.XX:=CX;
  COORD1.YY:=YMAX;
  IF AXE3NEG
   THEN COORD2.YY:=0
   ELSE COORD2.YY:=CY;
  TRDROITE(COORD1,COORD2,1,0);
  IF ABS(MULY/MULX)>(96/140)
   THEN NBPOINTS:=96/ABS(MULY)
   ELSE NBPOINTS:=140/ABS(MULX);
  COORD2.XX:=CX+TRUNC(NBPOINTS*MULX);
  IF MULX<0
   THEN COORD2.XX:=COORD2.XX+1;
  COORD2.YY:=CY+TRUNC(NBPOINTS*MULY)+1;
  IF AXE2NEG
   THEN
    BEGIN
      COORD1.XX:=XMAX-COORD2.XX;
      COORD1.YY:=YMAX-COORD2.YY
    END
   ELSE
    BEGIN
      COORD1.XX:=CX;
      COORD1.YY:=CY
    END;
  TRDROITE(COORD1,COORD2,1,0);
END;
(* ---------------------------------- *)
PROCEDURE TRPLAN(NROPLAN:INTEGER);
(* TRacer un PLAN *)

BEGIN
  (*$R TURTLEGRAPHICS *)
  IF NBSOMMET[NROplan]=0
   THEN (* plan non dessinnable *)
   ELSE
    BEGIN
      FOR I:=1 TO NBSOMMET[NROPLAN]
       DO PASSERA2DIM(PLAN3[NROPLAN,I],PLAN2[NROPLAN,I]);
      IF NBSOMMET[NROPLAN]=4
       THEN
        BEGIN
          TRDROITE(PLAN2[NROPLAN,1],PLAN2[NROPLAN,2],1,0);
          TRDROITE(PLAN2[NROPLAN,2],PLAN2[NROPLAN,3],1,0);
          TRDROITE(PLAN2[NROPLAN,3],PLAN2[NROPLAN,4],1,0);
          TRDROITE(PLAN2[NROPLAN,4],PLAN2[NROPLAN,1],1,0);
        END
       ELSE
        BEGIN
          TRDROITE(PLAN2[NROPLAN,1],PLAN2[NROPLAN,2],1,0);
          TRDROITE(PLAN2[NROPLAN,2],PLAN2[NROPLAN,3],1,1);
          TRDROITE(PLAN2[NROPLAN,3],PLAN2[NROPLAN,4],1,1);
```

```
            TRDROITE(PLAN2[NROPLAN,4],PLAN2[NROPLAN,1],1,1);
            TRDROITE(PLAN2[NROPLAN,2],PLAN2[NROPLAN,5],1,0);
            TRDROITE(PLAN2[NROPLAN,5],PLAN2[NROPLAN,6],1,0);
            TRDROITE(PLAN2[NROPLAN,6],PLAN2[NROPLAN,1],1,0);
          END
    END
 END;
(* ------------------------------------ *)
PROCEDURE DETUNITE(NRO:INTEGER;VAR UNITE:REAL);
(* DETerminer l'UNITE pour que tous les sommets du plan soient visibles *

VAR I,BINF,BSUP:INTEGER;
    MINIMUM:REAL;
(* ------------------------------- *)
PROCEDURE CHOIXUNITE(P1,P2,P3:REAL);
(* CHOIX d'une UNITE pour que un point en 3 dimensions soit visible *)

CONST MULTI=0.8;
BEGIN
  IF P1=0
  THEN
   IF P2=0
   THEN
    IF P3=0
    THEN UNITE:=MULTI*AXEX
    ELSE UNITE:=(MULTI*AXEY)/P3
   ELSE
    IF P3=0
    THEN
     BEGIN
       UNITE:=(MULTI*AXEX)/(P2*MULX);
       IF ABS(P2*MULY*UNITE)>AXEY
       THEN UNITE:=(MULTI*AXEY)/(P2*MULY)
     END
    ELSE
     BEGIN
       UNITE:=(MULTI*AXEX)/(MULX*P2);
       IF ABS(UNITE*((MULY*P2)+P3))>AXEY
       THEN UNITE:=(MULTI*AXEY)/((MULY*P2)+P3)
     END
  ELSE
   IF P2=0
   THEN
    IF P3=0
     THEN UNITE:=(MULTI*AXEX)/P1
     ELSE
      BEGIN
        UNITE:=(MULTI*AXEX)/P1;
        IF ABS(UNITE*P3)>AXEY
        THEN UNITE:=(MULTI*AXEY)/P3
      END
    ELSE
     IF P3=0
     THEN
      BEGIN
        UNITE:=(MULTI*AXEY)/(P2*MULY);
        IF ABS(UNITE*((P2*MULX)+P1))>AXEX
        THEN UNITE:=(MULTI*AXEX)/(P1+(P2*MULX));
      END
     ELSE
      BEGIN
        IF (P1+(P2*MULX))=0
         THEN UNITE:=1000
         ELSE UNITE:=(MULTI*AXEX)/(P1+(MULX*P2));
        IF ABS((P3+(MULY*P2))*UNITE)>AXEY
```

```
        THEN UNITE:=(MULTI*AXEY)/(P3+(MULY*P2))
      END;
 UNITE:=ABS(UNITE);
END;
(* ---------------------------------- *)

BEGIN
  MINIMUM:=30000;
  IF NBSOMMET[NRO]<>0
   THEN
    BEGIN
      IF NBSOMMET[NRO]=4
       THEN BEGIN
              BINF:=1;BSUP:=4
            END
       ELSE BEGIN
              BINF:=1;BSUP:=6
            END;
      FOR I:=BINF TO BSUP
       DO
        BEGIN
          CHOIXUNITE(PLAN3[NRO,I].P1,
                     PLAN3[NRO,I].P2,
                     PLAN3[NRO,I].P3);
          IF UNITE<MINIMUM THEN MINIMUM:=UNITE
        END;
    END;
  UNITE:=MINIMUM;
END;
(* ---------------------------------- *)
(* les procedures AXNEG1PL,AXNEG2PL,AXNEG3PL
   servent a positionner l'indicateur pour savoir s'il faut representer
   ou non la partie negative des axes,respectivement pour 1,2 ou 3 plans *

PROCEDURE AXNEG1PL(NROPLAN:INTEGER);
BEGIN
  AXE1NEG:=FALSE;
  AXE2NEG:=FALSE;
  AXE3NEG:=FALSE;
  IF AX1NEG[NROPLAN]
  THEN AXE1NEG:=TRUE;
  IF AX2NEG[NROPLAN]
  THEN AXE2NEG:=TRUE;
  IF AX3NEG[NROPLAN]
  THEN AXE3NEG:=TRUE;
END;
(* ------------------------------ *)
PROCEDURE AXNEG2PL(NROPL1,NROPL2:INTEGER);
BEGIN
  AXE1NEG:=FALSE;
  AXE2NEG:=FALSE;
  AXE3NEG:=FALSE;
  IF AX1NEG[NROPL1] OR AX1NEG[NROPL2]
  THEN AXE1NEG:=TRUE;
  IF AX2NEG[NROPL1] OR AX2NEG[NROPL2]
  THEN AXE2NEG:=TRUE;
  IF AX3NEG[NROPL1] OR AX3NEG[NROPL2]
  THEN AXE3NEG:=TRUE;
END;
(* ------------------------------ *)
PROCEDURE AXNEG3PL;
BEGIN
  AXE1NEG:=FALSE;
  AXE2NEG:=FALSE;
  AXE3NEG:=FALSE;
```

```
  IF AX1NEG[1] OR AX1NEG[2] OR AX1NEG[3]
  THEN AXE1NEG:=TRUE;
  IF AX2NEG[1] OR AX2NEG[2] OR AX2NEG[3]
  THEN AXE2NEG:=TRUE;
  IF AX3NEG[1] OR AX3NEG[2] OR AX3NEG[3]
  THEN AXE3NEG:=TRUE;
END;
(* ------------------------------- *)
PROCEDURE CHOIXORIENTATION(NROPL1,NROPL2:INTEGER);
(* CHOIX de l'ORIENTATION pour que la superposition de 2 plans s'effectuent
   dans un "bon" systeme de reference *)

VAR I:INTEGER;
    TROUVE:BOOLEAN;
BEGIN
  TROUVE:=FALSE;
  I:=1;
  WHILE (I<=6) AND (NOT TROUVE)
  DO BEGIN
       IF  (ORIENTATION[NROPL1,I])
       AND (ORIENTATION[NROPL2,I])
        THEN BEGIN
               TROUVE:=TRUE;
               MULX:=ORI[I,1];
               MULY:=ORI[I,2]
             END
        ELSE I:=I+1;
     END;
  IF NOT TROUVE
   THEN BEGIN
          MULX:=-1/2;
          MULY:=-3/4
        END
END;
(* ------------------------------- *)
PROCEDURE QUELLEORIENTATION;
(* meme but que CHOIXORIENTATION , mais pour 3 plans simultanement *)

VAR I:INTEGER;
    TROUVE:BOOLEAN;
BEGIN
  TROUVE:=FALSE;
  I:=1;
  WHILE (I<=6) AND (NOT TROUVE)
  DO BEGIN
       IF  (ORIENTATION[1,I])
       AND (ORIENTATION[2,I])
       AND (ORIENTATION[3,I])
        THEN BEGIN
               TROUVE:=TRUE;
               MULX:=ORI[I,1];
               MULY:=ORI[I,2]
             END
        ELSE I:=I+1;
     END;
  IF NOT TROUVE
   THEN BEGIN
          MULX:=-1/2;
          MULY:=-3/4
        END
END;
(* ------------------------------- *)
PROCEDURE INTERSECTION(P1,P2:POINT);
(* transforme 2 points "logiques" en points "physiques" et les joint
   P1 et P2 sont les extremites d'un segment d'INTERSECTION *)
```

```
VAR C1,C2:COORDONNEE;
BEGIN
  PASSERA2DIM(P1,C1);
  PASSERA2DIM(P2,C2);
  TRDROITE(C1,C2,11,0)
END;
(* ------------------------------ *)
PROCEDURE INIORIENTATION;
(* INItialisation de la matrice booleenne ORIENTATION *)

VAR I,J:INTEGER;
BEGIN
  FOR I:=1 TO 3
   DO FOR J:=1 TO 6
       DO ORIENTATION[I,J]:=TRUE
END;
(* ---------------------------- *)
PROCEDURE INITORI;
(* INItialisation de la matrice ORI (valeurs de MULX et MULY pour chaque
   orientation. *)

BEGIN
  ORI[1,1]:=-2/3;
  ORI[1,2]:=-2/3;
  ORI[2,1]:=2/3;
  ORI[2,2]:=-2/3;
  ORI[3,1]:=-2/3;
  ORI[3,2]:=-1/3;
  ORI[4,1]:=2/3;
  ORI[4,2]:=-1/3;
  ORI[5,1]:=-1/3;
  ORI[5,2]:=-2/3;
  ORI[6,1]:=1/3;
  ORI[6,2]:=-2/3;
END;
(* ------------------------------- *)
PROCEDURE UNPLAN;
(* rappel des parametres : (A1,B1,C1,D1:REAL;
                            NRO:INTEGER)
   represente un plan de coefficients a1,b1,c1,d1 *)

BEGIN
  AX1NEG[NRO]:=FALSE;
  AX2NEG[NRO]:=FALSE;
  AX3NEG[NRO]:=FALSE;
  QUELPLAN(A1,B1,C1,D1,NRO);
  DETUNITE(NRO,UNITE);
  INITTURTLE;
  AXNEG1PL(NRO);
  TRAXES;
  TRPLAN(NRO);
END;
(* ------------------------------ *)
PROCEDURE INTER2PL(N1,N2:INTEGER);
(* presente la superposition de 2 plans *)

BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  INTERPLAN(N1,N2);
  DETUNITE(N1,UNI[N1]);
  DETUNITE(N2,UNI[N2]);
  IF UNI[N1]<UNI[N2]
   THEN UNITE:=UNI[N1]
```

```
   ELSE UNITE:=UNI[N2];
  INITTURTLE;
  AXNEG2PL(N1,N2);
  CHOIXORIENTATION(N1,N2);
  TRAXES;
  TRPLAN(N1);
  TRPLAN(N2);
  INTERSECTION(INTER1,INTER2);
END;
(* ------------------------------ *)
PROCEDURE INTER3PL;
(* presente la superposition de 3 plans *)

BEGIN
  INITTURTLE;
  DETUNITE(1,UNI[1]);
  DETUNITE(2,UNI[2]);
  DETUNITE(3,UNI[3]);
  IF UNI[1]<UNI[2]
   THEN IF UNI[1]<UNI[3]
          THEN UNITE:=UNI[1]
          ELSE UNITE:=UNI[3]
   ELSE IF UNI[2]<UNI[3]
          THEN UNITE:=UNI[2]
          ELSE UNITE:=UNI[3];
  AXNEG3PL;
  QUELLEORIENTATION;
  TRAXES;
  TRPLAN(1);
  TRPLAN(2);
  TRPLAN(3);
  INTERSECTION(PTINTER[1,1],PTINTER[1,2]);
  INTERSECTION(PTINTER[2,1],PTINTER[2,2]);
  INTERSECTION(PTINTER[3,1],PTINTER[3,2]);
END;
(* ------------------------------ *)
PROCEDURE PTINTAZERO;
(* remise a zero du tableau de memorisation des points d'intersection *)

VAR I,J:INTEGER;
BEGIN
  FOR I:=1 TO 3
   DO FOR J:=1 TO 2
       DO BEGIN
            PTINTER[I,J].P1:=0;
            PTINTER[I,J].P2:=0;
            PTINTER[I,J].P3:=0;
          END;
END;
(* ------------------------------ *)
PROCEDURE INTAZERO;
(* remise a zero des 2 variables utilisees pour ranger les points
   d'intersection *)

BEGIN
  INTER1.P1:=0;INTER1.P2:=0;INTER1.P3:=0;
  INTER2.P1:=0;INTER2.P2:=0;INTER2.P3:=0;
END;
(* ------------------------------ *)
PROCEDURE DEUXPLANS;
(* gere la presentation d'un groupe de 2 plans
      - presentation de chacun des 2 plans
      - presentation de leur superposition   *)

BEGIN
```

```
  INIORIENTATION;
  UNPLAN(A1,B1,C1,D1,NRO1);
  SUITE;
  NBSOMMET[NRO1]:=4;
  UNPLAN(A2,B2,C2,D2,NRO2);
  SUITE;
  NBSOMMET[NRO2]:=4;
  INTAZERO;
  INTER2PL(NRO1,NRO2);
END;
(* ------------------------------ *)
PROCEDURE SYST33;
(* gere la presentation d'un groupe de 3 plans
      - presentation du 1er et du 2eme,puis de leur superposition
      - idem avec les plans 1 et 3
      - idem avec les plans 2 et 3
      - presentation simultanee des 3 plans   *)

BEGIN
  INIORIENTATION;
  PTINTAZERO;
  FOR I:=1 TO 3
  DO BEGIN
       AX1NEG[I]:=FALSE;
       AX2NEG[I]:=FALSE;
       AX3NEG[I]:=FALSE;
     END;
  FOR NRO:=1 TO 3
  DO BEGIN
       TEXTMODE;
       IF MODE=0
        THEN BEGIN
               AA[NRO]:=MATF[NRO,1].NUM/MATF[NRO,1].DEN;
               BB[NRO]:=MATF[NRO,2].NUM/MATF[NRO,2].DEN;
               CC[NRO]:=MATF[NRO,3].NUM/MATF[NRO,3].DEN;
               DD[NRO]:=MATF[NRO,4].NUM/MATF[NRO,4].DEN
             END
        ELSE BEGIN
               AA[NRO]:=MATR[NRO,1];
               BB[NRO]:=MATR[NRO,2];
               CC[NRO]:=MATR[NRO,3];
               DD[NRO]:=MATR[NRO,4]
             END;
     END;
  DEUXPLANS(1,2,AA[1],BB[1],CC[1],DD[1],
                AA[2],BB[2],CC[2],DD[2]);
  SUITE;
  DEUXPLANS(1,3,AA[1],BB[1],CC[1],DD[1],
                AA[3],BB[3],CC[3],DD[3]);
  SUITE;
  DEUXPLANS(2,3,AA[2],BB[2],CC[2],DD[2],
                AA[3],BB[3],CC[3],DD[3]);
  SUITE;
  INTAZERO;
  INTERPLAN(1,2);
  PTINTER[1,1]:=INTER1;
  PTINTER[1,2]:=INTER2;
  INTAZERO;
  INTERPLAN(1,3);
  PTINTER[2,1]:=INTER1;
  PTINTER[2,2]:=INTER2;
  INTAZERO;
  INTERPLAN(2,3);
  PTINTER[3,1]:=INTER1;
  PTINTER[3,2]:=INTER2;
```

```
  INTER3PL;
  SUITE;
END;
(* ------------------------------ *)
BEGIN
END.
```

```
*****************************************
*                                       *
*  U N I T S   D ' A F F I C H A G E S  *
*                                       *
*****************************************
```

* AFFICHAGE : ELLE COMPREND LES PROCEDURES D'AFFICHAGE A L'ECRAN (MENUS,MESSAGES,TEXTES).
ELLE EST UTILISEE PAR "COORDI".
(voir pp 83-90).

* AFF3 : ELLE COMPREND UNE PROCEDURE D'AFFICHAGE DE TEXTES EXPLICATIFS A L'ECRAN.
ELLE EST UTILISEE PAR "DEMO3".
(voir pp 91-95).

```
(*$S+,N+*)
UNIT AFFICHAGE;

INTERFACE
(*======*)

PROCEDURE AFFMENU(NROMENU,NBCHOIX:INTEGER;
                  VAR REPONSE:INTEGER);

PROCEDURE AFFMESSAGE(NROMESSAGE:INTEGER);

PROCEDURE AFFTEXTE(NROTEXTE:INTEGER);

(* ------------------------------ *)
IMPLEMENTATION
(*==========*)

PROCEDURE AFFMENU;
(* rappel des parametres :
     (NROMENU,NBCHOIX:INTEGER;
      VAR REPONSE:INTEGER);
   AFFiche le MENU correspondant a l'initialisation NROMENU et offrant
   NBCHOIX.
   saisit le choix de l'utilisateur et apres validation le range dans
   REPONSE.
*)


VAR I:INTEGER;
    MENU:ARRAY[1..5,1..2] OF STRING[40];
    TITRE:ARRAY[1..2] OF STRING[40];
    CAR:CHAR;



  (* chaque initialisation de menu a lieu en 2 parties :
       - le titre  (INITITi)
       - les choix  (INIMENUi)
  *)

  PROCEDURE INITIT1;
  BEGIN
    TITRE[1]:=' *** M E N U   P R I N C I P A L ***';
    TITRE[2]:=' *** IL Y A 5 CHOIX POSSIBLES :  ***';
  END;

  PROCEDURE INITIT2;
  BEGIN
    TITRE[1]:=' DESIREZ-VOUS VOIR LES INSTRUCTIONS ';
    TITRE[2]:='   A L''INTRODUCTION DES SYSTEMES ? ';
  END;

  PROCEDURE INITIT3;
  BEGIN
    TITRE[1]:=' ***   IL Y A 4 POSSIBILITES DE  ***';
    TITRE[2]:=' *** PRESENTATION DE LA SOLUTION ***';
  END;

  PROCEDURE INITIT4;
  BEGIN
    TITRE[1]:=' *** V O U S   P O U V E Z       ***';
    TITRE[2]:=' ***       M A I N T E N A N T : ***';
  END;
```

```
PROCEDURE INITIT5;
BEGIN
  TITRE[1]:='***     DESIREZ-VOUS CHOISIR    ***';
  TITRE[2]:='***     L''EXERCICE VOUS-MEME?   ***';
END;

PROCEDURE INITIT6;
BEGIN
  TITRE[1]:=' L''ORDINATEUR ATTEND VOTRE REPONSE:';
  TITRE[2]:='          VEUILLEZ PRESSER:          ';
END;

PROCEDURE INITIT7;
BEGIN
  TITRE[1]:=' ***    COMME VOTRE REPONSE EST   ***';
  TITRE[2]:=' ***      EXACTE VOUS POUVEZ :    ***';
END;

PROCEDURE INITIT8;
BEGIN
  TITRE[1]:=' ***    COMME VOTRE REPONSE EST   ***';
  TITRE[2]:=' ***    INEXACTE,VOUS POUVEZ :    ***';
END;

PROCEDURE INIMENU1;
BEGIN
  MENU[1,1]:=' L''ORDINATEUR RESOUD LES PRO-';
  MENU[1,2]:=' BLEMES QUE VOUS LUI PROPOSEZ.';
  MENU[2,1]:=' L''ORDINATEUR VERIFIE VOS ';
  MENU[2,2]:=' REPONSES A UN PROBLEME.';
  MENU[3,1]:=' L''ORDINATEUR FAIT LA DEMONSTRATION';
  MENU[3,2]:=' DES DIFFERENTS CAS DE SYSTEMES 3*3.';
  MENU[4,1]:=' IDEM POUR LES SYSTEMES 2*2.';
  MENU[4,2]:='';
  MENU[5,1]:=' VOUS POUVEZ QUITTER CE PROGRAMME.';
  MENU[5,2]:='';
END;

PROCEDURE INIMENU2;
BEGIN
  MENU[1,1]:='  SI OUI.                                    ';
  MENU[1,2]:='                                            ';
  MENU[2,1]:='  SI NON.                                   ';
  MENU[2,2]:='                                            ';
  MENU[3,1]:='';
  MENU[3,2]:='';
  MENU[4,1]:='';
  MENU[4,2]:='';
  MENU[5,1]:='';
  MENU[5,2]:='';
end;

PROCEDURE INIMENU3;
BEGIN
  MENU[1,1]:='L''ORDINATEUR DONNE UNIQUEMENT';
  MENU[1,2]:='LA SOLUTION NUMERIQUE.';
  MENU[2,1]:='L''ORDINATEUR DONNE EN PLUS LA';
  MENU[2,2]:='RESOLUTION GRAPHIQUE.(2*2 OU 3*3).';
  MENU[3,1]:='IL MONTRE EN PLUS LE CHEMINEMENT';
  MENU[3,2]:='DE L''ELIMINATION NUMERIQUE.';
  MENU[4,1]:='IL MONTRE EN PLUS LE CHEMINEMENT';
  MENU[4,2]:='GRAPHIQUE.(2*2 OU 3*3)';
  MENU[5,1]:='';
  MENU[5,2]:='';
END;
```

```
PROCEDURE INIMENU4;
BEGIN
  MENU[1,1]:='PROPOSER UN AUTRE EXERCICE.';
  MENU[1,2]:='';
  MENU[2,1]:='REPARTIR DE L''EXERCICE PRECEDENT';
  MENU[2,2]:='ET MODIFIER CERTAINES EQUATIONS';
  MENU[3,1]:='RETOURNER AU MENU PRINCIPAL.';
  MENU[3,2]:='';
  MENU[4,1]:='';
  MENU[4,2]:='';
  MENU[5,1]:='';
  MENU[5,2]:='';
END;

PROCEDURE INIMENU5;
BEGIN
  MENU[1,1]:='SI VOUS VOULEZ INTRODUIRE';
  MENU[1,2]:='VOTRE PROPRE SYSTEME.';
  MENU[2,1]:='SI VOUS VOULEZ RESOUDRE';
  MENU[2,2]:='UN EXERCICE PROPOSE PAR L''ORDINATEUR';
  MENU[3,1]:='';
  MENU[3,2]:='';
  MENU[4,1]:='';
  MENU[4,2]:='';
  MENU[5,1]:='';
  MENU[5,2]:='';
END;

PROCEDURE INIMENU6;
BEGIN
  MENU[1,1]:='SI LE SYSTEME ADMET UNE';
  MENU[1,2]:='      SOLUTION UNIQUE.';
  MENU[2,1]:='SI LE SYSTEME EST INDETERMINE.';
  MENU[2,2]:='';
  MENU[3,1]:='SI LE SYSTEME EST IMPOSSIBLE.';
  MENU[3,2]:='';
  MENU[4,1]:='';
  MENU[4,2]:='';
  MENU[5,1]:='';
  MENU[5,2]:='';
END;

PROCEDURE INIMENU7;
BEGIN
  MENU[1,1]:='VOIR UNE DES SOLUTIONS.';
  MENU[1,2]:='';
  MENU[2,1]:='CONTINUER CE GENRE D''EXERCICE.';
  MENU[2,2]:='';
  MENU[3,1]:='RETOURNER AU MENU PRINCIPAL.';
  MENU[3,2]:='';
  MENU[4,1]:='';
  MENU[4,2]:='';
  MENU[5,1]:='';
  MENU[5,2]:='';
END;

PROCEDURE INIMENU8;
BEGIN
  MENU[1,1]:='RECOMMENCER CET EXERCICE.';
  MENU[1,2]:='';
  MENU[2,1]:='EN VOIR LA SOLUTION.';
  MENU[2,2]:='';
  MENU[3,1]:='CHANGER D''EXERCICE.';
  MENU[3,2]:='';
```

```
    MENU[4,1]:='RETOURNER AU MENU PRINCIPAL.';
    MENU[4,2]:='';
    MENU[5,1]:='';
    MENU[5,2]:='';
  END;

  PROCEDURE VALIDATION(BINF,BSUP:INTEGER;
                       CHOIX:CHAR);
  (* lit un caractere et tant qu'il n'est pas compris entre les bornes
     BINF et BSUP envoie un message d'erreur *)

 BEGIN
    IF ((ORD(CHOIX)-48)<BINF)
    OR ((ORD(CHOIX)-48)>BSUP)
     THEN
       BEGIN
         GOTOXY(0,21);
         WRITELN ('VOTRE CHOIX DOIT ETRE COMPRIS');
         WRITELN ('ENTRE ',BINF,' ET ',BSUP);
         GOTOXY(10,23);
         WRITE ('VOTRE CHOIX (',BINF,' A ',BSUP,')?');
         READ (KEYBOARD,CAR);
         VALIDATION (BINF,BSUP,CAR)
       END
     ELSE
       REPONSE:=ORD(CHOIX)-48
  END;

BEGIN
  CASE NROMENU OF
   1:BEGIN INITIT1;INIMENU1 END;
   2:BEGIN INITIT2;INIMENU2 END;
   3:BEGIN INITIT3;INIMENU3 END;
   4:BEGIN INITIT4;INIMENU4 END;
   5:BEGIN INITIT5;INIMENU5 END;
   6:BEGIN INITIT6;INIMENU6 END;
   7:BEGIN INITIT7;INIMENU7 END;
   8:BEGIN INITIT8;INIMENU8 END;
  END;
  IF (NROMENU=2) OR (NROMENU=5)
   THEN GOTOXY(0,10)  (* dans ces 2 cas ce menu vient en-dessous d'un texte
   ELSE PAGE(OUTPUT);
  WRITELN(TITRE[1]);
  WRITELN(TITRE[2]);
  FOR I:=1 TO NBCHOIX
   DO BEGIN
        WRITELN('');
        WRITELN(I,'-->',MENU[I,1]);
        WRITELN('    ',MENU[I,2]);
      END;
  GOTOXY(0,21);
  WRITELN('VOTRE CHOIX ? (1-',NBCHOIX,')');
  READ(KEYBOARD,CAR);
  VALIDATION(1,NBCHOIX,CAR);
END;

(*************************************)

PROCEDURE AFFMESSAGE;
(* rappel des parametres
      (NROMESSAGE:INTEGER)
   AFFiche un MESSAGE (cad un texte tres court) a l'ecran *)

VAR I:INTEGER;
    MESSAGE:ARRAY[0..8] OF STRING[40];
```

```
(* l'initialisation de ces messages a lieu dans les procedures
      -INICi pour le rappel du choix
      -INIERi pour les messages d'erreur
      -INICORRECTi pour le message en cas de reponse correcte
*)

PROCEDURE INIC1;
BEGIN
  FOR I:=0 TO 8
  DO MESSAGE[I]:='';
  MESSAGE[1]:='     ****************************';
  MESSAGE[2]:='     * *  C H O I X  NRO 1  * *';
  MESSAGE[3]:='     ****************************';
  MESSAGE[5]:=' L''ORDINATEUR RESOUD LES PROBLEMES';
  MESSAGE[6]:='      QUE VOUS LUI PROPOSEZ.';
  MESSAGE[7]:=' ************************************';
END;

PROCEDURE INIC2;
BEGIN
  FOR I:=0 TO 8
  DO MESSAGE[I]:='';
  MESSAGE[1]:='     ****************************';
  MESSAGE[2]:='     * *  C H O I X  NRO 2 * *';
  MESSAGE[3]:='     ****************************';
  MESSAGE[5]:='L''ORDINATEUR VERIFIE VOS REPONSES';
  MESSAGE[6]:='************************************';
END;

PROCEDURE INIER1;
BEGIN
  FOR I:=0 TO 8
  DO MESSAGE[I]:='';
  MESSAGE[1]:='************************************';
  MESSAGE[2]:='* ATTENTION , VOUS AVEZ COMMIS  *';
  MESSAGE[3]:='*    UNE ERREUR !                *';
  MESSAGE[4]:='*                                *';
  MESSAGE[5]:='* CE SYSTEME N''ADMET PAS UNE    *';
  MESSAGE[6]:='* SOLUTION UNIQUE.               *';
  MESSAGE[7]:='************************************';
END;

PROCEDURE INIER2;
BEGIN
  FOR I:=0 TO 8
  DO MESSAGE[I]:='';
  MESSAGE[1]:='************************************';
  MESSAGE[2]:='* VOUS AVEZ COMMIS UNE ERREUR! *';
  MESSAGE[3]:='*                              *';
  MESSAGE[4]:='* CE SYSTEME ADMET BIEN UNE    *';
  MESSAGE[5]:='* SOLUTION UNIQUE,MAIS PAS     *';
  MESSAGE[6]:='* CELLE QUE VOUS AVEZ TROUVEE! *';
  MESSAGE[7]:='************************************';
END;

PROCEDURE INIER3;
BEGIN
  FOR I:=0 TO 8
  DO MESSAGE[I]:='';
  MESSAGE[1]:='************************************';
  MESSAGE[2]:='* VOUS AVEZ COMMIS UNE ERREUR!  *';
  MESSAGE[3]:='*                               *';
  MESSAGE[4]:='* CE SYSTEME N''EST PAS          *';
  MESSAGE[5]:='*                INDETERMINE !  *';
```

```
    MESSAGE[6]:='************************************';
  END;

  PROCEDURE INIER4;
  BEGIN
    FOR I:=0 TO 8
    DO MESSAGE[I]:='';
    MESSAGE[1]:='************************************';
    MESSAGE[2]:='* VOUS AVEZ COMMIS UNE ERREUR!   *';
    MESSAGE[3]:='*                                 *';
    MESSAGE[4]:='* IL S''AGIT BIEN D''UN SYSTEME    *';
    MESSAGE[5]:='* INDETERMINE , MAIS L''ORDRE     *';
    MESSAGE[6]:='* D''INDETERMINATION EST FAUX!    *';
    MESSAGE[7]:='************************************';
  END;

  PROCEDURE INIER5;
  BEGIN
    FOR I:=0 TO 8
    DO MESSAGE[I]:='';
    MESSAGE[1]:='************************************';
    MESSAGE[2]:='* VOUS AVEZ COMMIS UNE ERREUR!   *';
    MESSAGE[3]:='*                                 *';
    MESSAGE[4]:='* CE SYSTEME N''EST PAS           *';
    MESSAGE[5]:='*                  IMPOSSIBLE     *';
    MESSAGE[6]:='************************************';
  END;

  PROCEDURE INIEXACT;
  BEGIN
    FOR I:=0 TO 8
    DO MESSAGE[I]:='';
    MESSAGE[1]:='************************************';
    MESSAGE[2]:='*     !!! FELICITATIONS !!!      *';
    MESSAGE[3]:='*                                 *';
    MESSAGE[4]:='* VOTRE REPONSE EST CORRECTE.     *';
    MESSAGE[5]:='************************************';
  END;

BEGIN
  CASE NROMESSAGE OF
   1:INIC1;
   2:INIC2;
   3:INIER1;
   4:INIER2;
   5:INIER3;
   6:INIER4;
   7:INIER5;
   8:INIEXACT;
  END;
  PAGE(OUTPUT);
  FOR I:=0 TO 8
  DO WRITELN(MESSAGE[I])
END;

(************************************)

PROCEDURE AFFTEXTE;
(* rappel des parametres
     (NROTEXTE:INTEGER);
   AFFiche un TEXTE a l'ecran *)

VAR I:INTEGER;
    TEXTE:ARRAY[0..22] OF STRING[40];
```

```
(* les initialisations de textes ont lieu dans les procedures INITi *)

PROCEDURE INIT1;
BEGIN
  FOR I:=0 TO 22
  DO TEXTE[I]:='';
  TEXTE[1]:='    SYSTEMES D''EQUATIONS LINEAIRES';
  TEXTE[2]:='    ******************************';
  TEXTE[5]:='BONJOUR, CE PROGRAMME RESOUD ET  ';
  TEXTE[6]:='TENTE DE MONTRER LE MECANISME DES';
  TEXTE[7]:='SYSTEMES D''EQUATIONS LINEAIRES.';
  TEXTE[9]:='DANS LE CAS DES SYSTEMES 2*2 ET 3*3';
  TEXTE[10]:='IL EST POSSIBLE D''OBTENIR LEUR ';
  TEXTE[11]:='REPRESENTATION GRAPHIQUE.';
  TEXTE[14]:='  2 REMARQUES GENERALES : ';
  TEXTE[15]:='  ---------------------- ';
  TEXTE[16]:='  --> NE PRESSEZ JAMAIS LES TOUCHES';
  TEXTE[17]:='    ''CTRL''  ET  ''RESET''  ';
  TEXTE[18]:='  --> LORSQUE VOUS AVEZ TERMINE ';
  TEXTE[19]:='      DE LIRE UN ECRAN : ';
  TEXTE[20]:='  SOIT VOUS SELECTIONNEZ UN NRO';
  TEXTE[21]:='  SOIT VOUS PRESSEZ UNE TOUCHE QCQ';
END;

PROCEDURE INIT2;
BEGIN
  FOR I:=0 TO 22
  DO TEXTE[I]:='';
  TEXTE[1]:='VEUILLEZ INTRODUIRE VOTRE SYSTEME ';
  TEXTE[2]:='EN TENANT COMPTE DES REMARQUES.';
  TEXTE[4]:='1 --> LES VARIABLES SONT APPELEES';
  TEXTE[5]:='      ''X'' ,SUIVI D''UN INDICE';
  TEXTE[7]:='2 --> LE NOMBRE MAX D''EQUATIONS = 5';
  TEXTE[9]:='3 --> LE NOMBRE MAX D''INCONNUES = 5';
  TEXTE[11]:='4 --> TOUTES LES VARIABLES SONT A';
  TEXTE[12]:='      GAUCHE DU SIGNE ''=''.LEUR';
  TEXTE[13]:='      ORDRE N''A PAS D''IMPORTANCE.';
  TEXTE[15]:='5 --> LE TERME INDEPENDANT EST TOUJOURS';
  TEXTE[16]:='      SITUE A DROITE DU SIGNE ''=''.';
  TEXTE[18]:='6 --> LES COEFFICIENTS PEUVENT ETRE';
  TEXTE[19]:='      ENTIERS,FRACTIONNAIRES OU REELS.';
  TEXTE[21]:='7 --> VOUS POUVEZ INSERER DES BLANCS.';
  TEXTE[22]:='      EXEMPLE : 2/3 X2 -0.3X1 = 4 ';
END;

PROCEDURE INIT3;
BEGIN
  FOR I:=0 TO 22
  DO TEXTE[I]:='';
  TEXTE[1]:='REMARQUES SUR LES COEFFICIENTS REELS';
  TEXTE[2]:='------------------------------------';
  TEXTE[3]:='--> LE SEPARATEUR ENTRE LES PARTIES';
  TEXTE[4]:='DECIMALES ET ENTIERES EST ''.'' OU '','''';
  TEXTE[6]:='--> LORSQUE LA PARTIE ENTIERE EST ';
  TEXTE[7]:='NULLE ON PEUT L''OMETTRE.';
  TEXTE[8]:='EXPLE : .34 EQUIVAUT A 0.34';
  TEXTE[10]:='REMARQUES SUR LES FRACTIONS';
  TEXTE[11]:='---------------------------';
  TEXTE[13]:='--> LE NUMERATEUR ET LE DENOMINATEUR';
  TEXTE[14]:='SONT DES ENTIERS,SEPARES PAR LA ';
  TEXTE[15]:='BARRE DE FRACTION ''/''.';
  TEXTE[17]:='--> LE SIGNE DOIT ETRE PLACE DEVANT';
  TEXTE[18]:='LE NUMERATEUR.(-3/4 ET PAS 3/-4)';
  TEXTE[20]:='--> LA FRACTION DOIT SE TROUVER ';
  TEXTE[21]:='AVANT LE ''X'' ET PAS APRES.';
```

```
    TEXTE[22]:='EXPLE : 1/3 X2 ET NON PAS X2/3';
    END;

  PROCEDURE INIT4;
  BEGIN
    FOR I:=0 TO 22
    DO TEXTE[I]:='';
    TEXTE[1]:='REMARQUES SUR LES CORRECTIONS';
    TEXTE[2]:='-------------------------';
    TEXTE[3]:='SI VOUS AVEZ COMMIS UNE ERREUR DANS';
    TEXTE[4]:='LA LIGNE QUE VOUS INTRODUISEZ :';
    TEXTE[6]:='-->VOUS POUVEZ L''EFFACER EN PRESSANT';
    TEXTE[7]:='   LA TOUCHE (<-).TOUTE LA LIGNE S''EF-';
    TEXTE[8]:='   FACERA ET VOUS LA REINTRODUISEZ.';
    TEXTE[10]:='-->VOUS POUVEZ AUSSI RECOPIER LE';
    TEXTE[11]:='   COUPLE (COEFFICIENT,INCONNUE) ERRONE';
    TEXTE[15]:='DE TOUTE FACON , A LA FIN DE L''INTRO-';
    TEXTE[16]:='DUCTION , IL VOUS SERA ENCORE POSSIBLE';
    TEXTE[17]:='DE CORRIGER TOUTES LES EQUATIONS.';
  END;

  PROCEDURE INIT5;
  BEGIN
    FOR I:=0 TO 22
    DO TEXTE[I]:='';
    TEXTE[2]:='**************************************';
    TEXTE[3]:='* * F I N   D U   T R A V A I L * *';
    TEXTE[4]:='**************************************';
    TEXTE[8]:='VOUS AVEZ DECIDE DE QUITTER CET OUTIL';
    TEXTE[9]:='NOUS ESPERONS QUE VOTRE TRAVAIL AVEC';
    TEXTE[10]:='LUI AURA ETE A LA FOIS AGREABLE ET';
    TEXTE[11]:='PROFITABLE.';
    TEXTE[15]:='     **********************';
    TEXTE[16]:='     *  A U   R E V O I R *';
    TEXTE[17]:='     **********************';
  END;

BEGIN
  CASE NROTEXTE OF
   1:INIT1;
   2:INIT2;
   3:INIT3;
   4:INIT4;
   5:INIT5;
  END;
  PAGE(OUTPUT);
  FOR I:=0 TO 22
  DO WRITELN(TEXTE[I]);
END;
(* ------------------------------- *)
BEGIN
END.
(* ------------------------------- *)
```

```
(*$S++,N+*)
UNIT AFF3;INTRINSIC CODE 18;


(* CETTE UNIT COMPREND LA PROCEDURE AFFTEXT3,QUI A EXACTEMENT LA MEME
   STRUCTURE QUE "AFFTEXTE" (VOIR DANS "AFFICHAGE").ELLE EST UTILISEE
   PAR LE PROGRAMME "DEMO3" POUR LES EXPLICATIONS CONCERNANT LES DIFFERENTS
   PLANS,LEURS SUPERPOSITIONS ... *)


INTERFACE
(*******)
PROCEDURE AFFTEXT3(NRO:INTEGER);

IMPLEMENTATION
(*************)
PROCEDURE AFFTEXT3;

VAR TEXTE:PACKED ARRAY [0..23] OF STRING[40];
    I:INTEGER;
    C:CHAR;

    PROCEDURE INITEXTE;
    BEGIN
      FOR I:=0 TO 23 DO TEXTE[I]:='';
    END;

    PROCEDURE INIT1;
    BEGIN
      TEXTE[6]:='          ************************';
      TEXTE[7]:='          *                      *';
      TEXTE[8]:='          *  DEMONSTRATION 3*3   *';
      TEXTE[9]:='          *                      *';
      TEXTE[10]:='          ************************';
      TEXTE[13]:='          ELLE COMPORTE 3 PARTIES : ';
      TEXTE[15]:='1 : PRESENTATION DE DIFFERENTS PLANS.';
      TEXTE[17]:='2 : PRESENTATION DE SUPERPOSITIONS ';
      TEXTE[18]:='    DE PLANS.';
      TEXTE[20]:='3 : PRESENTATION DES DIFFERENTS TYPES';
      TEXTE[21]:='    DE SYSTEMES 3*3.';
    END;


    PROCEDURE INIT2;
    BEGIN
      TEXTE[1]:=' 1ERE  PARTIE : PRESENTATION DES PLANS';
      TEXTE[2]:=' ************************************';
      TEXTE[4]:='UNE EQUATION " AX1 + BX2 + CX3 = D "';
      TEXTE[5]:='PEUT GENERALEMENT (SAUF EQUATION IMPOS-';
      TEXTE[6]:='SIBLE OU INDETERMINEE) SE REPRESENTER ';
      TEXTE[7]:='PAR UN PLAN.';
      TEXTE[9]:='SI LES 4 COEFFICIENTS SONT <> 0,NOUS ';
      TEXTE[10]:='POUVONS DETERMINER LE POINT D''INTERSEC-';
      TEXTE[11]:='TION DU PLAN AVEC CHACUN DES 3 AXES:';
      TEXTE[12]:='  I1=(D/A,0,0) SUR L''AXE X1';
      TEXTE[13]:='  I2=(0,D/B,0) SUR L''AXE X2';
      TEXTE[14]:='  I3=(0,0,D/C) SUR L''AXE X3';
      TEXTE[15]:='SI NOUS JOIGNONS I1 A I2 NOUS OBTENONS';
      TEXTE[16]:='UN SEGMENT DE LA DROITE D''INTERSECTION';
      TEXTE[17]:='DU PLAN A REPRESENTER ET DU PLAN HORI-';
      TEXTE[18]:='ZONTAL.EN FAISANT DE MEME AVEC I1 ET I3';
      TEXTE[19]:='NOUS SITUONS L''INTERSECTION AVEC LE';
      TEXTE[20]:='PLAN DE FACE.';
      TEXTE[21]:='CES 2 SEGMENTS DE DROITE SERVENT DE BASE';
      TEXTE[22]:='A LA REPRESENTATION DE CE TYPE DE PLAN';
```

```
END;

 PROCEDURE INIT3;
 BEGIN
   TEXTE[1]:='COMMENCONS PAR L''EXEMPLE SUIVANT:';
   TEXTE[3]:='              **********************';
   TEXTE[4]:='              *                     *';
   TEXTE[5]:='              * 2X1 + 2X2 + X3 =4 *';
   TEXTE[6]:='              *                     *';
   TEXTE[7]:='              **********************';
   TEXTE[8]:='                  I1=(2,0,0)';
   TEXTE[9]:='                  I2=(0,2,0)';
   TEXTE[10]:='                  I3=(0,0,4)';
   TEXTE[12]:='NOUS RELIONS LES POINTS I1 ET I2';
   TEXTE[13]:='AINSI QUE I2 ET I3.DE PLUS NOUS ELE-';
   TEXTE[14]:='VONS 2 PARALLELES A CES SEGMENTS A';
   TEXTE[15]:='PARTIR DES POINTS I2 ET I3.NOUS OBTE-';
   TEXTE[16]:='NONS AINSI UN QUADRILATERE QUI PERMET';
   TEXTE[17]:='DE "VOIR" QUEL PLAN REPRESENTE CETTE';
   TEXTE[18]:='EQUATION.VOYONS LE GRAPHIQUE :';
   TEXTE[20]:='NOUS VERRONS,GRACE AUX GRAPHIQUES ';
   TEXTE[21]:='SUIVANTS CE QUI CHANGE SI UN DES ';
   TEXTE[22]:='3 COEFFICIENTS A,B OU C EST <0';
 END;

 PROCEDURE INIT4;
 BEGIN
   TEXTE[5]:='SUR LES 3 DERNIERS SCHEMAS,NOUS AVONS';
   TEXTE[6]:='CONSTATE QUE PARFOIS LA POSITION DE ';
   TEXTE[7]:='L''AXE X2 EST MODIFIEE POUR PERMETTRE';
   TEXTE[8]:='UNE MEILLEURE VISUALISATION.';
   TEXTE[11]:='DANS LE MEME ORDRE D''IDEE,SI LA PARTIE';
   TEXTE[12]:='DU PLAN QUE NOUS REPRESENTONS SE SITUE';
   TEXTE[13]:='SOUS LE NIVEAU DE L''HORIZONTALE,NOUS';
   TEXTE[14]:='LA PROLONGEONS EGALEMENT AU-DESSUS DE';
   TEXTE[15]:='CE NIVEAU.';
   TEXTE[16]:='QUE SE PASSE-T-IL SI D=0? AX1+BX2+CX3=0';
   TEXTE[17]:='NOUS NE POUVONS PLUS TROUVER 3 POINTS';
   TEXTE[18]:='DISTINCTS I1,I2 ET I3.NOUS REPRESENTONS';
   TEXTE[19]:='EGALEMENT CE PLAN A PARTIR DE SA DROITE';
   TEXTE[20]:='D''INTERSECTION AVEC LE PLAN HORIZONTAL.';
   TEXTE[21]:='NOUS ELEVONS ALORS 2 SEGMENTS PARALLE-';
   TEXTE[22]:='LES POUR DONNER UNE IDEE DE L''INCLINAI-';
   TEXTE[23]:='SON DE CE PLAN.VOYONS 2 EXEMPLES : ';
 END;


 PROCEDURE INIT5;
 BEGIN
   TEXTE[2]:='PASSONS MAINTENANT AUX CAS OU L''UN DES';
   TEXTE[3]:='COEFFICIENTS A,B ET C EST = 0.';
   TEXTE[4]:='SUR LES GRAPHIQUES SUIVANTS,NOUS CONSTA-';
   TEXTE[5]:='TERONS QUE CE GENRE DE PLAN EST PARALLE-';
   TEXTE[6]:='LE A L''AXE CORRESPONDANT AU COEFFICIENT';
   TEXTE[7]:='NUL.';
   TEXTE[10]:=' - UN PLAN D''EQUATION "BX2 + CX3 = D"';
   TEXTE[11]:='   EST PARALLELE A L''AXE X1';
   TEXTE[13]:=' - UN PLAN D''EQUATION "AX1 + CX3 = D"';
   TEXTE[14]:='   EST PARALLELE A L''AXE X2';
   TEXTE[16]:=' - UN PLAN D''EQUATION "AX1 + BX2 = D"';
   TEXTE[17]:='   EST PARALLELE A L''AXE X3';
 END;


 PROCEDURE INIT6;
```

```
BEGIN
  TEXTE[2]:='NOUS AVONS VU QUE SI UN DES COEFFICIENTS';
  TEXTE[3]:='A,B OU C EST NUL,LE PLAN EST PARALLELE';
  TEXTE[4]:='A L''AXE CORRESPONDANT.';
  TEXTE[7]:='SI 2 DE CES COEFFICIENTS SONT = 0,ALORS';
  TEXTE[8]:='LE PLAN SERA PARALLELE AUX 2 AXES COR-';
  TEXTE[9]:='RESPONDANTS.';
  TEXTE[11]:=' -''AX1 = D : CE PLAN EST DONC PARALLELE';
  TEXTE[12]:='   AUX AXES X2 ET X3,C''EST A DIRE QU''IL';
  TEXTE[13]:='   EST PARALLELE AU PLAN DEBOUT.';
  TEXTE[15]:=' - BX2 = D : CE PLAN EST DONC PARALLELE';
  TEXTE[16]:='   AUX AXES X1 ET X3,C''EST A DIRE QU''IL';
  TEXTE[17]:='   EST PARALLELE AU PLAN DE FACE.';
  TEXTE[19]:=' - CX3 = D : CE PLAN EST DONC PARALLELE';
  TEXTE[20]:='   AUX AXES X1 ET X2,C''EST A DIRE QU''IL';
  TEXTE[21]:='   EST PARALLELE AU PLAN HORIZONTAL.';
END;

PROCEDURE INIT7;
BEGIN
  TEXTE[1]:='NOUS PRESENTONS ENCORE QUELQUES PLANS ';
  TEXTE[2]:='PARTICULIERS.';
  TEXTE[4]:=' - SI D=0 ET UN DES COEFFICIENTS A,B OU';
  TEXTE[5]:='   C EGALEMENT,NOUS AURONS AFFAIRE A UN';
  TEXTE[6]:='   PLAN PASSANT PAR L''ORIGINE ET PARAL-';
  TEXTE[7]:='   LELE A UN AXE.SELON LA VALEUR DES 2 ';
  TEXTE[8]:='   COEFFICIENTS NON-NULS,CE TYPE DE PLAN';
  TEXTE[9]:='   "PIVOTE" AUTOUR DE CET AXE.';
  TEXTE[12]:=' - SI D=0 ET DEUX DES COEFFICIENTS A,B';
  TEXTE[13]:='   OU C SONT NULS,CE PLAN PASSE PAR L''O-';
  TEXTE[14]:='   RIGINE ET EST PARALLELE A 2 AXES : ';
  TEXTE[16]:='     * AX1=0 REPRESENTE LE PLAN DEBOUT';
  TEXTE[17]:='     * BX2=0 REPRESENTE LE PLAN DE FACE';
  TEXTE[18]:='     * CX3=0 REPRESENTE LE PLAN HORI-';
  TEXTE[19]:='       ZONTAL';
END;

PROCEDURE INIT8;
BEGIN
  TEXTE[1]:='2EME PARTIE:SUPERPOSITION DE PLANS .';
  TEXTE[2]:='************************************';
  TEXTE[4]:='IL Y A 3 POSSIBILITES :';
  TEXTE[5]:=' 1 LES 2 PLANS SONT CONFONDUS';
  TEXTE[6]:='   NOUS REMARQUERONS QUE DANS CE CAS LES';
  TEXTE[7]:='   2 EQUATIONS SONT MULTIPLES L''UNE DE';
  TEXTE[8]:='   DE L''AUTRE.';
  TEXTE[10]:=' 2 LES 2 PLANS SONT PARALLELES';
  TEXTE[11]:='   NOUS REMARQUERONS QUE DANS CE CAS LES';
  TEXTE[12]:='   2 EQUATIONS SONT MULTIPLES L''UNE DE';
  TEXTE[13]:='   DE L''AUTRE,AU TERME INDEPENDANT PRES.';
  TEXTE[15]:=' 3 LES 2 PLANS SONT SECANTS';
  TEXTE[16]:='   NOUS DETAILLERONS LES DIFFERENTES ';
  TEXTE[17]:='   POSSIBILITES PLUS LOIN.';
  TEXTE[20]:='REM:NOUS NE TENONS PAS COMPTE DES ';
  TEXTE[21]:='EQUATIONS IMPOSSIBLES ET INDETERMINEES';
END;

PROCEDURE INIT9;
BEGIN
  TEXTE[2]:='SI LES 2 PLANS SONT SECANTS,IL EXISTE ';
  TEXTE[3]:='UNE DROITE D''INTERSECTION ENTRE EUX.';
  TEXTE[4]:='EN GENERAL,NOUS REPRESENTONS CELLE-CI';
  TEXTE[5]:='PAR UN SEGMENT QUI LIE LE POINT D''INTER-';
  TEXTE[6]:='SECTION SITUE SUR LE PLAN HORIZONTAL ET ';
  TEXTE[7]:='CELUI SITUE SUR LE PLAN DE FACE.';
```

```
  TEXTE[9]:='NOUS VERRONS QUE PARFOIS IL NOUS FAUDRA';
  TEXTE[10]:='AGGRANDIR LES PORTIONS DE PLAN PRESEN-';
  TEXTE[11]:='TEES POUR POUVOIR VISUALISER CES 2';
  TEXTE[12]:='POINTS.';
  TEXTE[14]:='DE PLUS COMME CERTAINS PLANS N''ONT PAS';
  TEXTE[15]:='D''INTERSECTION AVEC LE PLAN HORIZONTAL';
  TEXTE[16]:='ET/OU LE PLAN DE FACE,CES 2 POINTS';
  TEXTE[17]:='N''EXISTENT PAS : NOUS VERRONS CELA SUR ';
  TEXTE[18]:='QUELQUES EXEMPLES.';
END;

PROCEDURE INIT10;
BEGIN
  TEXTE[1]:='3EME PARTIE:SYSTEMES 3*3 COMPLETS .';
  TEXTE[2]:='**********************************';
  TEXTE[3]:='IL Y A 3 POSSIBILITES : ';
  TEXTE[5]:=' 1 LE SYSTEME ADMET UNE SOLUTION UNIQUE';
  TEXTE[7]:='   CETTE SOLUTION EST LE POINT D''INTER-';
  TEXTE[8]:='   SECTION ENTRE LES DROITES D''INTERSEC-';
  TEXTE[9]:='   TION DES PLANS 2 A 2.';
  TEXTE[10]:='    NOUS ALLONS EN VOIR 2 EXEMPLES.';
  TEXTE[13]:=' 2 LE SYSTEME EST IMPOSSIBLE.';
  TEXTE[15]:='   NOUS DETAILLERONS PLUS LOIN.';
  TEXTE[18]:=' 3 LE SYSTEME EST INDETERMINE.';
  TEXTE[20]:='   NOUS DETAILLERONS PLUS LOIN.';
END;

PROCEDURE INIT11;
BEGIN
  TEXTE[2]:='NOUS ALLONS VOIR MAINTENANT POURQUOI UN';
  TEXTE[3]:='SYSTEME 3*3 PEUT ETRE IMPOSSIBLE :';
  TEXTE[6]:='  - LES 3 PLANS SONT PARALLELES ';
  TEXTE[9]:='  - 2 PLANS SONT CONFONDUS ET LE 3EME ';
  TEXTE[10]:='    LEUR EST PARALLELE.';
  TEXTE[13]:='  - 2 PLANS SONT PARALLELES ET LE 3EME';
  TEXTE[14]:='    LEUR EST SECANT.';
  TEXTE[17]:='  - LES 3 PLANS SONT TELS QUE LEURS ';
  TEXTE[18]:='    INTERSECTIONS 2 A 2 SONT DES DROITES';
  TEXTE[19]:='    PARALLELES.';
  TEXTE[21]:='VOYONS UN EXEMPLE DE CES 2 DERNIERS CAS.';
END;

PROCEDURE INIT12;
BEGIN
  TEXTE[1]:='NOUS TERMINONS EN EXAMINANT POURQUOI UN';
  TEXTE[2]:='SYSTEME 3*3 PEUT ETRE INDETERMINE :';
  TEXTE[4]:=' - IL PEUT ETRE INDETERMINE D''ORDRE 2 ';
  TEXTE[5]:='   LES 3 PLANS SONT CONFONDUS.';
  TEXTE[7]:=' - IL PEUT ETRE INDETERMINE D''ORDRE 1';
  TEXTE[8]:='   ET DANS CE CAS TOUS LES POINTS SOLU-';
  TEXTE[9]:='   TIONS SONT SITUES SUR UNE DROITE.';
  TEXTE[11]:='    2 POSSIBILITES :';
  TEXTE[12]:='      - 2 PLANS SONT CONFONDUS ET LE ';
  TEXTE[13]:='        3EME LEUR EST SECANT.';
  TEXTE[14]:='        (NB:CELA REVIENT EN FAIT A LA';
  TEXTE[15]:='         SUPERPOSITION DE 2 PLANS)';
  TEXTE[17]:='      - LES 3 PLANS SONT SITUES DE MA-';
  TEXTE[18]:='        NIERE TELLE QUE LEURS DROITES';
  TEXTE[19]:='        D''INTERSECTION SONT CONFONDUES';
  TEXTE[21]:='        NOUS ALLONS VOIR 1 EXEMPLE DU 1ER';
  TEXTE[22]:='        CAS ET 2 DU SECOND.';
END;

PROCEDURE INIT13;
BEGIN
```

```
      TEXTE[11]:='*********************************';
      TEXTE[12]:='*                               *';
      TEXTE[13]:='* FIN DE LA DEMONSTRATION 3*3 *';
      TEXTE[14]:='*                               *';
      TEXTE[15]:='*********************************';
    END;


(* DEBUT DE AFFTEXT3 *)
BEGIN
  INITEXTE;
  CASE NRO OF
   1:INIT1;
   2:INIT2;
   3:INIT3;
   4:INIT4;
   5:INIT5;
   6:INIT6;
   7:INIT7;
   8:INIT8;
   9:INIT9;
   10:INIT10;
   11:INIT11;
   12:INIT12;
   13:INIT13;
  END;
  PAGE(OUTPUT);
  FOR I:=1 TO 23
   DO WRITELN(TEXTE[I]);
  IF NRO<>13 THEN READ(KEYBOARD,C);
END;

(* ------------------------------ *)
BEGIN
END.
```

PROGRAMMES

* COORDI : CE PROGRAMME REALISE LE PROGRAMME DECRIT DANS LE MEMOIRE,A L'EXCEPTION DES REPRESENTATIONS GRAPHIQUES POUR LES RAISONS EXPLIQUEES.
(voir pp 97-103).

* DEMO2 : C'EST LE PROGRAMME DE DEMONSTRATION DES SYSTEMES 2*2.
(voir pp 104-109).

* DEMO3 : C'EST LE PROGRAMME DE DEMONSTRATION DES SYSTEMES 3*3.
(voir pp 110-112).

* GRAPH : CE PROGRAMME PERMET DE REPRESENTER DES DROITES (ET DES SYSTEMES 2*2),AINSI QUE DES PLANS (ET DES SYSTEMES 3*3).
(voir pp 113-117)

```
(*$S+,N*)
PROGRAM COORDINATEUR;
USES APPLESTUFF,UTILUNIT,SYSUNIT,
     (* MANQUE DE PLACE SI ON CHARGE TOUTES LES UNITS
     TURTLEGRAPHICS,DESSIN22,
     DECLAR33,CHOIXPLAN,INTER33,DESSIN33,
     *)
     (*$U APPLE2:LIB.LIBRARY*)
     INTROUNIT,AFFICHAGE;

VAR RETOURMENU,RETOURVERIF,DEJACHOISI,CORRECT,FIN:BOOLEAN;
(* ------------------------------ *)
PROCEDURE INTROSYSTEME;
(**********************)
VAR REPONSE:INTEGER;
BEGIN
  AFFMENU(2,2,REPONSE);
  IF REPONSE=1
   THEN BEGIN
          AFFTEXTE(2);
          SUITE;
          AFFTEXTE(3);
          SUITE;
          AFFTEXTE(4);
          SUITE;
        END;
  PAGE(OUTPUT);
  INTRODUCTION(SYSTEME);
  WHILE SYSTVIDE
  DO BEGIN
       PAGE(OUTPUT);
       WRITELN('VOUS N''AVEZ PAS INTRODUIT DE SYSTEME.');
       INTRODUCTION(SYSTEME)
     END;
  CONTINUER:=TRUE;
  WHILE CONTINUER
  DO BEGIN
       DEMANDER;
       IF CONTINUER THEN CORSYS
     END;
  DETERMINERMODE;
  MODEDEPART:=MODE;
  IF MODE=0 THEN SIMPLIMAT;
  EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
END;
(* ------------------------------- *)
PROCEDURE CORRECTION;
(********************)
BEGIN
  CONTINUER:=TRUE;
  WHILE CONTINUER
  DO BEGIN
       CORSYS;
       DEMANDER
     END;
  DETERMINERMODE;
  MODEDEPART:=MODE;
  IF MODE=0 THEN SIMPLIMAT;
  EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
END;
(* ------------------------------- *)
PROCEDURE CREATION;
(******************)
VAR TAMPON:INTEGER;
```

```
    PROCEDURE MESERREUR;
    BEGIN
      BIP;
      WRITELN;
      WRITELN('VOTRE REPONSE EST INCORRECTE.VEUILLEZ');
      WRITELN('ENTRER UN NOMBRE ENTRE 1 ET 5.');
      READ(KEYBOARD,CAR);
    END;

BEGIN
  (*$R APPLESTUFF*)
  PAGE(OUTPUT);
  MODE:=0;MODEDEPART:=0;
  FOR I:=1 TO 5 DO OK[I]:=TRUE;
  WRITELN('COMBIEN D''EQUATIONS DESIREZ-VOUS? (1-5)');
  READ(KEYBOARD,CAR);
  WHILE (ORD(CAR)<49) OR (ORD(CAR)>53)
  DO MESERREUR;
  NBLIGNE:=ORD(CAR)-48;
  WRITELN('COMBIEN D''INCONNUES DESIREZ-VOUS? (1-5)');
  WHILE NOT KEYPRESS DO RANDOMIZE;
  READ(KEYBOARD,CAR);
  WHILE (ORD(CAR)<49) OR (ORD(CAR)>53)
  DO MESERREUR;
  INDMAX:=ORD(CAR)-48;
  FOR I:=1 TO NBLIGNE
  DO FOR J:=1 TO INDMAX+1
     DO BEGIN
          MAT[I,J].NUM:=RANDOM MOD 6;
          MAT[I,J].DEN:=1;
        END;
  EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
  SIMPLIMAT;
END;
(* ---------------------------------- *)
PROCEDURE PATIENCE;
(******************)
BEGIN
  PAGE(OUTPUT);
  GOTOXY(0,10);
  WRITELN('UN PEU DE PATIENCE , SVP ');
  WRITELN('   NOUS CALCULONS LA SOLUTION .');
END;
(* ---------------------------------- *)
PROCEDURE CHOIXSOLUTION;
(************************)
VAR REPONSE:INTEGER;
BEGIN
  AFFMENU(3,4,REPONSE);
  PATIENCE;
  CASE REPONSE OF
   1:RESOLUTION(1,0,0);
   2:RESOLUTION(1,1,0);
   3:RESOLUTION(1,0,1);
   4:RESOLUTION(1,1,1);
  END;
END;
(* ------------------------------ *)
PROCEDURE VSUNIQUE;
(******************)
  PROCEDURE COMPFRACTION;
  BEGIN
    I:=1;CORRECT:=TRUE;
```

```
    DO IF  (XF[I].NUM = SFUNIQUE[I].NUM)
       AND (XF[I].DEN = SFUNIQUE[I].DEN)
        THEN I:=I+1
        ELSE BEGIN
               CORRECT:=FALSE;
               AFFMESSAGE(4);
               SUITE
             END
  END;
  (* ---------------------------- *)
  PROCEDURE COMPREEL;
  (******************)
  BEGIN
    I:=1;CORRECT:=TRUE;
    WHILE (I<=INDMAX) AND (CORRECT)
    DO IF  (XR[I].SOLUTION = SRUNIQUE[I].SOLUTION)
        THEN I:=I+1
        ELSE BEGIN
               CORRECT:=FALSE;
               AFFMESSAGE(4);
               SUITE
             END
  END;
  (* ---------------------------- *)
BEGIN
  IF TYPESOL<>UNIQUE
  THEN BEGIN
         CORRECT:=FALSE;
         AFFMESSAGE(3);
         SUITE
       END
  ELSE BEGIN
         PAGE(OUTPUT);
         INTRODUCTION(SOLUNIQUE);
         CONTINUER:=TRUE;
         WHILE CONTINUER DO CORSOLUNIQUE;
         SIMPLISUNIQUE;
         IF MODE=0
          THEN IF SUNFRAC
                 THEN COMPFRACTION
                 ELSE BEGIN
                        CORRECT:=FALSE;
                        AFFMESSAGE(4);
                        SUITE
                      END
          ELSE IF SUNFRAC
                 THEN BEGIN
                        CORRECT:=FALSE;
                        AFFMESSAGE(4);
                        SUITE
                      END
                 ELSE COMPREEL
       END
END;
(* ---------------------------- *)
PROCEDURE VSINDETERMINEE;
(************************)
VAR CAR:CHAR;
BEGIN
  IF TYPESOL <> INDET
   THEN BEGIN
          AFFMESSAGE(5);
          SUITE
        END
```

```
            WRITELN;
            WRITELN;
            WRITELN('QUEL EST L''ORDRE D''INDETERMINATION?');
            READ(KEYBOARD,CAR);
            IF ORDREIND = ORD(CAR)-48
             THEN CORRECT:=TRUE
             ELSE BEGIN
                    AFFMESSAGE(6);
                    SUITE;
                    CORRECT:=FALSE
                  END;
          END
END;
(* ------------------------------ *)
PROCEDURE VSIMPOSSIBLE;
(**********************)
BEGIN
  IF TYPESOL<>IMPOS
   THEN BEGIN
          CORRECT:=FALSE;
          AFFMESSAGE(7);
          SUITE
        END
   ELSE CORRECT:=TRUE
END;
(* ------------------------------ *)
PROCEDURE ACCEPTERSOLUTION;
(**************************)
VAR REPONSE:INTEGER;
BEGIN
  AFFMENU(6,3,REPONSE);
  CASE REPONSE OF
   1:VSUNIQUE;
   2:VSINDETERMINEE;
   3:VSIMPOSSIBLE
  END;
END;
(* ------------------------- *)
PROCEDURE SOLCORRECT;
(********************)
VAR REPONSE:INTEGER;
BEGIN
  RETOURVERIF:=FALSE;
  WHILE NOT RETOURVERIF
   DO
    BEGIN
      AFFMENU(7,3,REPONSE);
      CASE REPONSE OF
       1:BEGIN
           MODE:=MODEDEPART;
           EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
           CHOIXSOLUTION;
         END;
       2:BEGIN
           RETOURVERIF:=TRUE;
           DEJACHOISI:=FALSE;
         END;
       3:BEGIN
           RETOURMENU:=TRUE;
           RETOURVERIF:=TRUE
         END;
      END;
    END;
END;
```

```
PROCEDURE SOLINCORRECT;
(**********************)
VAR REPONSE:INTEGER;
BEGIN
  RETOURVERIF:=FALSE;
  WHILE NOT RETOURVERIF
   DO
    BEGIN
     AFFMENU(8,4,REPONSE);
     CASE REPONSE OF
      1:BEGIN
          DEJACHOISI:=TRUE;
          RETOURVERIF:=TRUE;
        END;
      2:BEGIN
          MODE:=MODEDEPART;
          EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
          CHOIXSOLUTION;
        END;
      3:BEGIN
          RETOURVERIF:=TRUE;
          DEJACHOISI:=FALSE
        END;
      4:BEGIN
          RETOURMENU:=TRUE;
          RETOURVERIF:=TRUE
        END;
     END;
   END;
END;
(* ------------------------------ *)
PROCEDURE RESPROBLEME;
(*********************)
VAR REPONSE:INTEGER;
BEGIN
  RETOURMENU:=FALSE;
  DEJACHOISI:=FALSE;
  WHILE NOT RETOURMENU
   DO
    BEGIN
      IF NOT DEJACHOISI
       THEN
        BEGIN
          AFFMESSAGE(1);
          INTROSYSTEME
        END
       ELSE
        BEGIN
          PAGE(OUTPUT);
          MODE:=0;
          IMPMAT(MAT,MREEL);
          CORRECTION;
          EGALERMAT(NBLIGNE,INDMAX+1,MAT,INTER,MREEL,INTREEL);
        END;
      CHOIXSOLUTION;
      AFFMENU(4,3,REPONSE);
      CASE REPONSE OF
       1:DEJACHOISI:=FALSE;
       2:DEJACHOISI:=TRUE;
       3:RETOURMENU:=TRUE
      END;
    END;
END;
(* ------------------------------ *)
PROCEDURE VERIFICATIONPROBLEME;
```

```
(*****************************)
VAR REPONSE:INTEGER;
BEGIN
  RETOURMENU:=FALSE;
  DEJACHOISI:=FALSE;
  WHILE NOT RETOURMENU
   DO
    BEGIN
      IF NOT DEJACHOISI
       THEN
        BEGIN
          AFFMESSAGE(2);
          AFFMENU(5,2,REPONSE);
          CASE REPONSE OF
           1:INTROSYSTEME;(*CHOIXEXERCICE;*)
           2:BEGIN
               CREATION;(*DONNEREXERCICE*)
               IMPMAT(INTER,INTREEL);
               SUITE
             END;
          END;
          PATIENCE;
          RESOLUTION(0,0,0);
        END;
      ACCEPTERSOLUTION;
      IF CORRECT
       THEN BEGIN
              AFFMESSAGE(8);
              SUITE;
              SOLCORRECT
            END
       ELSE SOLINCORRECT;
    END;
 END;
 (* --------------------------- *)
PROCEDURE DEMO2;
(***************)
VAR CAR:CHAR;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(5,10);
  WRITELN ('DEMO2');
  READ(KEYBOARD,CAR);
END;
(* ------------------------------ *)
PROCEDURE DEMO3;
(***************)
VAR CAR:CHAR;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(5,10);
  WRITELN ('DEMO3');
  READ(KEYBOARD,CAR);
END;
(* ------------------------------ *)
PROCEDURE FINPROGRAMME;
(**********************)
BEGIN
  FIN:=TRUE;
  AFFTEXTE(5);(* NB PAS DE READ *)
  FOR I:=1 TO 3000
   DO;
END;
(* ------------------------------ *)
PROCEDURE MENUPRINCIPAL;
```

```
(**********************)
VAR REPONSE:INTEGER;
  BEGIN
    AFFMENU(1,5,REPONSE);
    CASE REPONSE OF
     1:RESPROBLEME;
     2:VERIFICATIONPROBLEME;
     3:DEMO3;
     4:DEMO2;
     5:FINPROGRAMME
    END
  END;
(* ------------------------------ *)
BEGIN
  AFFTEXTE(1);
  INIINTRODUCTION;
  SUITE;
  FIN:=FALSE;
  WHILE NOT FIN
   DO MENUPRINCIPAL
END.
```

```
(*$S+,N+*)
PROGRAM DEMO2;
USES APPLESTUF,UTILUNIT,TURTLEGRAPHICS,
     DESSIN22;

SEGMENT PROCEDURE AFFTEXT2(NRO:INTEGER);
VAR TEXTE:PACKED ARRAY [0..23] OF STRING[40];
    I:INTEGER;

    PROCEDURE INITEXTE;
    BEGIN
      FOR I:=0 TO 23 DO TEXTE[I]:='';
    END;

    PROCEDURE INIT1;
    BEGIN
      TEXTE[6]:='          ************************';
      TEXTE[7]:='          *                      *';
      TEXTE[8]:='          *  DEMONSTRATION 2*2   *';
      TEXTE[9]:='          *                      *';
      TEXTE[10]:='          ************************';
      TEXTE[13]:='          ELLE COMPORTE 2 PARTIES : ';
      TEXTE[15]:='1 : PRESENTATION DE DIFFERENTES DROITES.';
      TEXTE[17]:='2 : PRESENTATION DE DIFFERENTS SYSTEMES.';
    END;


    PROCEDURE INIT2;
    BEGIN
      TEXTE[1]:='UNE EQUATION DU TYPE AX1+BX2=C PEUT EN ';
      TEXTE[2]:='GENERAL SE REPRESENTER PAR UNE DROITE.';
      TEXTE[3]:='LA POSITION DE CETTE DROITE DEPEND BIEN';
      TEXTE[4]:='SUR DE LA VALEUR DES 3 COEFFICIENTS.';
      TEXTE[6]:='IL Y A 2 CAS TRES PARTICULIERS OU NOUS';
      TEXTE[7]:='NE SAURONS PAS REPRESENTER L''EQUATION ';
      TEXTE[8]:='PAR UNE DROITE.';
      TEXTE[9]:='SI A=0 ET B=0 ALORS,SELON LA VALEUR DE C';
      TEXTE[10]:='NOUS AURONS :';
      TEXTE[11]:='    - SOIT 0X1+0X2=0  (SI C=0)';
      TEXTE[12]:='    - SOIT 0X1+0X2<>0  (SI C<>0)';
      TEXTE[14]:='DANS LE PREMIER CAS IL S''AGIT D''UNE ';
      TEXTE[15]:='EQUATION INDETERMINEE.TOUS LES POINTS ';
      TEXTE[16]:='(X1,X2) EN SONT SOLUTIONS PUISQUE L''EX-';
      TEXTE[17]:='PRESSION 0X1+0X2=0 EST VRAIE POUR TOUT ';
      TEXTE[18]:='COUPLE (X1,X2)';
      TEXTE[20]:='INVERSEMENT L''EQUATION 0X1+0X2<>0 EST ';
      TEXTE[21]:='IMPOSSIBLE.AUCUNE VALEUR DE (X1,X2) NE ';
      TEXTE[22]:='CONVIENDRA JAMAIS.';
    END;

    PROCEDURE INIT3;
    BEGIN
      TEXTE[2]:='EXAMINONS MAINTENANT LES AUTRES CAS:';
      TEXTE[4]:='      COMMENCONS PAR L''EQUATION';
      TEXTE[6]:='          ****************';
      TEXTE[7]:='          *               *';
      TEXTE[8]:='          * 2X1 + 0X2 =3 *';
      TEXTE[9]:='          *               *';
      TEXTE[10]:='          ****************';
      TEXTE[13]:='LA SOLUTION NUMERIQUE DE CETTE EQUATION';
      TEXTE[14]:='EST  " X1 = 3/2 ".';
      TEXTE[16]:='AINSI ,QUELLE QUE SOIT LA VALEUR DE X2,';
      TEXTE[17]:='LA VALEUR DE X1 SERA TOUJOURS 3/2.';
      TEXTE[18]:='VOYONS CELA SUR LE GRAPHIQUE :';
    END;
```

```
PROCEDURE INIT4;
BEGIN
  TEXTE[4]:='COMME NOUS L''AVONS CONSTATE SUR LE';
  TEXTE[5]:='SCHEMA CETTE DROITE EST PARALLELE A';
  TEXTE[6]:='L''AXE X2.LA DISTANCE ENTRE L''AXE ET';
  TEXTE[7]:='LA DROITE EST CONSTANTE ET ELLE VAUT';
  TEXTE[8]:=' "C/A".';
  TEXTE[11]:='AINSI LORSQUE C=0, L''EQUATION AX1=0';
  TEXTE[12]:='REPRESENTE L''AXE X2 LUI-MEME.';
END;

PROCEDURE INIT5;
BEGIN
  TEXTE[1]:='PASSONS MAINTENANT A UN AUTRE TYPE';
  TEXTE[2]:='D''EQUATION.';
  TEXTE[4]:='          ******************';
  TEXTE[5]:='          *                *';
  TEXTE[6]:='          * 0X1 + 1X2 =-2 *';
  TEXTE[7]:='          *                *';
  TEXTE[8]:='          ******************';
  TEXTE[10]:='LA SOLUTION NUMERIQUE DE CETTE EQUATION';
  TEXTE[11]:='EST "X2 = -2".';
  TEXTE[13]:='AINSI,QUELLE QUE SOIT LA VALEUR DE X1,';
  TEXTE[14]:='LA VALEUR DE X2 SERA TOUJOURS -2.';
  TEXTE[15]:='CONSTATONS CELA SUR LE GRAPHIQUE :';
END;


PROCEDURE INIT6;
BEGIN
  TEXTE[2]:='CETTE DROITE EST DONC PARALLELE A ';
  TEXTE[3]:='L''AXE X1, A UNE DISTANCE CONSTANTE QUI';
  TEXTE[4]:='VAUT "C/B".';
  TEXTE[7]:='AINSI,SI C=0,L''EQUATION "BX2=0" ';
  TEXTE[8]:='REPRESENTE L''AXE X1 LUI-MEME.';
END;

PROCEDURE INIT7;
BEGIN
  TEXTE[1]:='PASSONS MAINTENANT AU TYPE D''EQUATION';
  TEXTE[2]:='OU AUCUN COEFFICIENT N''EST NUL.';
  TEXTE[3]:='PAR EXEMPLE :';
  TEXTE[4]:='          ****************';
  TEXTE[5]:='          *              *';
  TEXTE[6]:='          * 2X1 + 3X2 =6 *';
  TEXTE[7]:='          *              *';
  TEXTE[8]:='          ****************';
  TEXTE[10]:='POUR L''EXEMPLE CHOISI,LES POINTS (3,0)';
  TEXTE[11]:='ET (0,2) SONT SOLUTIONS.ILS SONT SITUES ';
  TEXTE[12]:='RESPECTIVEMENT SUR LES AXES X1 ET X2.';
  TEXTE[13]:='CE NE SONT EVIDEMMENT PAS LES SEULES ';
  TEXTE[14]:='SOLUTIONS:IL Y EN A UNE INFINITE.';
  TEXTE[15]:='VOICI QUELQUES EXEMPLES:';
  TEXTE[16]:='  (3/2,1)  (1,4/3)  (2,2/3) ...';
  TEXTE[18]:='COMME NOUS POUVONS LE VOIR SUR LE GRA-';
  TEXTE[19]:='PHIQUE SUIVANT,TOUS CES POINTS SONT ';
  TEXTE[20]:='SITUES SUR UNE MEME DROITE.';
END;

PROCEDURE INIT8;
BEGIN
  TEXTE[1]:='QUE SE PASSE-T-IL SI C=0 ?';
  TEXTE[2]:='L''EQUATION DEVIENT AX1+BX2=0.';
  TEXTE[3]:='VOYONS PAR EXEMPLE:';
```

```
  TEXTE[5]:='          ***************';
  TEXTE[6]:='          *             *';
  TEXTE[7]:='          * X1 - 2X2 =0 *';
  TEXTE[8]:='          *             *';
  TEXTE[9]:='          ***************';
  TEXTE[11]:='SI COMME TOUT A L''HEURE,NOUS CHERCHONS';
  TEXTE[12]:='LES POINTS D''INTERSECTION AVEC LES AXES';
  TEXTE[13]:='NOUS N''EN TROUVONS QU''UN SEUL : (0,0).';
  TEXTE[14]:='AFIN DE DETERMINER LA POSITION DE LA ';
  TEXTE[15]:='DROITE,NOUS POUVONS CHERCHER QUELQUES';
  TEXTE[16]:='AUTRES POINTS : PAR EXEMPLE :';
  TEXTE[17]:='  (2,1)  (-2,-1)  (1,1/2)  (-1,-1/2) ...';
  TEXTE[19]:='VERIFIONS LE SUR LE GRAPHIQUE :';
END;

PROCEDURE INIT9;
BEGIN
  TEXTE[1]:='EN RESUME,NOUS AVONS DONC :';
  TEXTE[3]:='  - SI A=0 ET B=0';
  TEXTE[4]:='     L''EQUATION EST 0=0  (INDETERMINEE)';
  TEXTE[5]:='                    OU 0<>0 (IMPOSSIBLE)';
  TEXTE[7]:='  - SI A=0 L''EQUATION EST BX2=C';
  TEXTE[8]:='     IL S''AGIT D''UNE DROITE PARALLELE';
  TEXTE[9]:='     A L''AXE X1 (SI C<>0) OU DE L''AXE X1';
  TEXTE[10]:='     LUI-MEME (SI C=0).';
  TEXTE[12]:='  - SI B=0 L''EQUATION EST AX1=C';
  TEXTE[13]:='     IL S''AGIT D''UNE DROITE PARALLELE';
  TEXTE[14]:='     A L''AXE X2 (SI C<>0) OU DE L''AXE X2';
  TEXTE[15]:='     LUI-MEME (SI C=0).';
  TEXTE[17]:='  - SI A<>0 ET B<>0';
  TEXTE[18]:='     L''EQUATION EST AX1+BX2=C';
  TEXTE[19]:='      * SI C<>0 IL S''AGIT D''UNE DROITE';
  TEXTE[20]:='         PASSANT PAR LES POINTS (C/A,0)';
  TEXTE[21]:='         ET (0,C/B).';
  TEXTE[22]:='      * SI C=0 IL S''AGIT D''UNE DROITE';
  TEXTE[23]:='         PASSANT PAR L''ORIGINE (0,0).';
END;

PROCEDURE INIT10;
BEGIN
  TEXTE[1]:='APRES AVOIR VU LES DIFFERENTS CAS DE ';
  TEXTE[2]:='DROITES, VOYONS CE QUI PEUT SE PASSER ';
  TEXTE[3]:='POUR UN SYSTEME 2*2:';
  TEXTE[4]:='  - SI UNE OU L''AUTRE DES 2 EQUATIONS';
  TEXTE[5]:='    EST IMPOSSIBLE (0<>0) ,LE SYSTEME ';
  TEXTE[6]:='    L''EST EGALEMENT.';
  TEXTE[7]:='  - SI LES DEUX EQUATIONS SONT INDETER-';
  TEXTE[8]:='    MINEES (0=0),LE SYSTEME L''EST EGA-';
  TEXTE[9]:='    MENT:TOUT COUPLE(X1,X2) EST SOLUTION';
  TEXTE[10]:='  - SI UNE DES 2 DROITES EST INDETERMI-';
  TEXTE[11]:='    NEE,LE SYSTEME EST INDETERMINE ET';
  TEXTE[12]:='    TOUS LES POINTS DE L''AUTRE DROITE';
  TEXTE[13]:='    SONT sOLUTIONS.';
  TEXTE[14]:='  - DANS TOUS LES AUTRES CAS,NOUS RE- ';
  TEXTE[15]:='    PRESENTONS LES 2 DROITES ET NOUS';
  TEXTE[16]:='    VOYONS DE QUOI IL S''AGIT:';
  TEXTE[17]:='       LES 2 DROITES SONT SECANTES  ';
  TEXTE[18]:='        --> LEUR POINT D''INTERSECTION';
  TEXTE[19]:='            EST LA SOLUTION UNIQUE.';
  TEXTE[20]:='       LES 2 DROITES SONT CONFONDUES  ';
  TEXTE[21]:='        --> LEURS POINTS SONT SOLUTIONS';
  TEXTE[22]:='       LES 2 DROITES SONT PARALLELES  ';
  TEXTE[23]:='        --> IL N''Y A PAS DE SOLUTION.';
END;
```

```
PROCEDURE INIT11;
BEGIN
  TEXTE[1]:='COMMENCONS PAR LE CAS OU LES DEUX ';
  TEXTE[2]:='DROITES SONT SECANTES:';
  TEXTE[5]:='  PRENONS L''EXEMPLE SUIVANT : ';
  TEXTE[7]:='         *****************         ';
  TEXTE[8]:='         *                 *';
  TEXTE[9]:='         * 2X1 - 4X2 = 5 *';
  TEXTE[10]:='         *  X1 + 3X2 = 0 *';
  TEXTE[11]:='         *                 *';
  TEXTE[12]:='         *****************';
  TEXTE[15]:='  LEUR POINT D''INTERSECTION VAUT :';
  TEXTE[16]:='            ( 3/2 , -1/2 )';
  TEXTE[18]:='  IL S''AGIT DE LA SOLUTION UNIQUE';
  TEXTE[19]:='  DE CE SYSTEME.  ';
END;

PROCEDURE INIT12;
BEGIN
  TEXTE[2]:='PASSONS AU CAS OU LES 2 DROITES SONT ';
  TEXTE[3]:='CONFONDUES : PRENONS L''EXEMPLE :';
  TEXTE[5]:='           ******************';
  TEXTE[6]:='           *                *';
  TEXTE[7]:='           * 3X1 - 6X2 = 9  *';
  TEXTE[8]:='           * -X1 + 2X2 = -3 *';
  TEXTE[9]:='           *                *';
  TEXTE[10]:='           ******************';
  TEXTE[12]:='CES DEUX EQUATIONS REPRESENTENT UNE ';
  TEXTE[13]:='SEULE ET MEME DROITE : LE SYSTEME EST';
  TEXTE[14]:='DONC INDETERMINE ET IL ADMET UNE INFINI-';
  TEXTE[15]:='TE DE SOLUTIONS (CE SONT TOUS LES ';
  TEXTE[16]:='POINTS DE LA DROITE).';
  TEXTE[19]:='NOUS POUVONS REMARQUER QUE LES 2 EQUA-';
  TEXTE[20]:='TIONS SONT MULTIPLES L''UNE DE L''AUTRE.';
END;

PROCEDURE INIT13;
BEGIN
  INITEXTE;
  TEXTE[2]:='NOUS ARRIVONS AU DERNIER CAS : LES 2 ';
  TEXTE[3]:='DROITES SONT PARALLELES.';
  TEXTE[4]:='PAR EXEMPLE : ';
  TEXTE[6]:='           *****************';
  TEXTE[7]:='           *               *';
  TEXTE[8]:='           *  X1 + 2X2 = 4 *';
  TEXTE[9]:='           * 3X1 + 6X2 = 6 *';
  TEXTE[10]:='           *               *';
  TEXTE[11]:='           *****************';
  TEXTE[14]:='COMME CES 2 DROITES N''ONT AUCUN POINT';
  TEXTE[15]:='COMMUN,CE SYSTEME N''ADMET EVIDEMMENT';
  TEXTE[16]:='PAS DE SOLUTION : IL S''AGIT D''UN SYS-';
  TEXTE[17]:='TEME IMPOSSIBLE.';
  TEXTE[19]:='NOUS POUVONS REMARQUER QUE LES 2 EQUATI-';
  TEXTE[20]:='ONS SONT MULTIPLES L''UNE DE L''AUTRE,AU';
  TEXTE[21]:='TERME INDEPENDANT PRES.';
END;

PROCEDURE INIT14;
BEGIN
  INITEXTE;
  TEXTE[10]:='   ***********************************';
  TEXTE[11]:='   *                                 *';
  TEXTE[12]:='   *  FIN DE LA DEMONSTRATION 2*2    *';
  TEXTE[13]:='   *                                 *';
  TEXTE[14]:='   ***********************************';
```

```
    END;

(* ---------------------------------- *)

(* DEBUT DE AFFTEXT2 *)

BEGIN
  INITEXTE;
  CASE NRO OF
   1:INIT1;
   2:INIT2;
   3:INIT3;
   4:INIT4;
   5:INIT5;
   6:INIT6;
   7:INIT7;
   8:INIT8;
   9:INIT9;
   10:INIT10;
   11:INIT11;
   12:INIT12;
   13:INIT13;
   14:INIT14;
  END;
  PAGE(OUTPUT);
  TEXTMODE;
  FOR I:=1 TO 23
   DO WRITELN(TEXTE[I]);
  IF NRO<>14 THEN SUITE;
END;

(* ------------------------------ *)


SEGMENT PROCEDURE UNEDROITE(P1,P2,P3:INTEGER);
VAR A,B,C:FRACTION;
BEGIN
  RECALC:=FALSE;
  NDR:=1;
  MODE:=0;
  A.NUM:=P1;
  B.NUM:=P2;
  C.NUM:=P3;
  A.DEN:=1;
  B.DEN:=1;
  C.DEN:=1;
  GR22(A,B,C,0,0,0);
  SUITE
END;

SEGMENT PROCEDURE DEUXDROITES(A1,B1,C1,A2,B2,C2:INTEGER);
VAR MF:MATRICE;
    MR:MATREEL;

BEGIN
  MODE:=0;
  MF[1,1].NUM:=A1;
  MF[1,2].NUM:=B1;
  MF[1,3].NUM:=C1;
  MF[2,1].NUM:=A2;
  MF[2,2].NUM:=B2;
  MF[2,3].NUM:=C2;
  MF[1,1].DEN:=1;
  MF[1,2].DEN:=1;
  MF[1,3].DEN:=1;
```

```
    MF[2,1].DEN:=1;
    MF[2,2].DEN:=1;
    MF[2,3].DEN:=1;
    SYST22(MF,MR);
    SUITE
END;

(* ------------------------ *)
(* BEGIN DU PROGRAMME DEMO 2 *)
(* ------------------------ *)

BEGIN
    AFFTEXT2(1);
    AFFTEXT2(2);
    AFFTEXT2(3);
    UNEDROITE(2,0,3);
    AFFTEXT2(4);
    AFFTEXT2(5);
    UNEDROITE(0,1,-2);
    AFFTEXT2(6);
    AFFTEXT2(7);
    UNEDROITE(2,3,6);
    AFFTEXT2(8);
    UNEDROITE(1,-2,0);
    AFFTEXT2(9);
    AFFTEXT2(10);
    AFFTEXT2(11);
    DEUXDROITES(2,-4,5,1,3,0);
    AFFTEXT2(12);
    DEUXDROITES(3,-6,9,-1,2,-3);
    AFFTEXT2(13);
    DEUXDROITES(1,2,4,3,6,6);
    AFFTEXT2(14);
END.
```

```
(*$S+,N*)
PROGRAM DEMO3;
USES APPLESTUFF,UTILUNIT,TURTLEGRAPHICS,
     DECLAR33,CHOIXPLAN,INTER33,DESSIN33,
     AFF3;

SEGMENT PROCEDURE COEFFICIENTS(A,B,C,D:REAL);
VAR PLUSMOINS:CHAR;
BEGIN
  WRITE('        ',ROUND(A),' X1 ');
  IF B<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:='+';
  WRITE(PLUSMOINS,ABS(ROUND(B)),' X2 ');
  IF C<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:='+';
  WRITE(PLUSMOINS,ABS(ROUND(C)),' X3 ');
  IF D<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:=' ';
  WRITELN('=',PLUSMOINS,ABS(ROUND(D)));
END;


SEGMENT PROCEDURE TRDEUXPLANS(A1,B1,C1,D1,
                        A2,B2,C2,D2:REAL);
BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('  SUPERPOSITION DES 2 PLANS');
  COEFFICIENTS(A1,B1,C1,D1);
  COEFFICIENTS(A2,B2,C2,D2);
  AA[1]:=A1;
  BB[1]:=B1;
  CC[1]:=C1;
  DD[1]:=D1;
  AA[2]:=A2;
  BB[2]:=B2;
  CC[2]:=C2;
  DD[2]:=D2;
  MULX:=-2/3;
  MULY:=-2/3;
  SUITE;
  DEUXPLANS(1,2,
            AA[1],BB[1],CC[1],DD[1],
            AA[2],BB[2],CC[2],DD[2]);
  SUITE
END;


SEGMENT PROCEDURE TRTROISPLANS(A1,B1,C1,D1,
                         A2,B2,C2,D2,
                         A3,B3,C3,D3:REAL);
VAR M1:MATRICE;
    M2:MATREEL;

BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('  PRESENTATION DU SYSTEME 3*3 :');
  COEFFICIENTS(A1,B1,C1,D1);
  COEFFICIENTS(A2,B2,C2,D2);
  COEFFICIENTS(A3,B3,C3,D3);
  MODE:=1;
  M2[1,1]:=A1;
```

```
  M2[1,2]:=B1;
  M2[1,3]:=C1;
  M2[1,4]:=D1;
  M2[2,1]:=A2;
  M2[2,2]:=B2;
  M2[2,3]:=C2;
  M2[2,4]:=D2;
  M2[3,1]:=A3;
  M2[3,2]:=B3;
  M2[3,3]:=C3;
  M2[3,4]:=D3;
  SUITE;
  SYST33(M1,M2);
END;

PROCEDURE TRUNPLAN(A,B,C,D:REAL);
BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('REPRESENTATION DE L''EQUATION :');
  COEFFICIENTS(A,B,C,D);
  SUITE;
  UNPLAN(A,B,C,D,1);
  SUITE;
END;


BEGIN
  AFFTEXT3(1);
  INITORI;
  MODE:=1;
  AFFTEXT3(2);
  AFFTEXT3(3);
  TRUNPLAN(2,2,1,4);
  TRUNPLAN(-2,2,1,4);
  TRUNPLAN(2,-2,1,4);
  TRUNPLAN(2,2,-1,4);
  TEXTMODE;AFFTEXT3(4);
  TRUNPLAN(1,2,1,0);
  TRUNPLAN(1,2,-1,0);
  TEXTMODE;AFFTEXT3(5);
  TRUNPLAN(0,1,1,4);
  TRUNPLAN(2,0,1,4);
  TRUNPLAN(1,-4,0,4);
  TEXTMODE;AFFTEXT3(6);
  TRUNPLAN(1,0,0,2);
  TRUNPLAN(0,1,0,-2);
  TRUNPLAN(0,0,1,2);
  TEXTMODE;AFFTEXT3(7);
  TRUNPLAN(2,1,0,0);
  TRUNPLAN(0,2,1,0);
  TRUNPLAN(2,0,1,0);
  TRUNPLAN(1,0,0,0);
  TRUNPLAN(0,1,0,0);
  TRUNPLAN(0,0,1,0);
  TEXTMODE;AFFTEXT3(8);
  TRDEUXPLANS(1,2,2,4,3,6,6,12);
  TRDEUXPLANS(1,2,2,4,3,6,6,6);
  TEXTMODE;AFFTEXT3(9);
  TRDEUXPLANS(3,6,4,12,6,4,3,12);
  TRDEUXPLANS(3,2,0,12,1,2,1,6);
  TRDEUXPLANS(2,3,4,12,6,4,3,12);
  TRDEUXPLANS(6,3,4,12,2,6,3,12);
  TRDEUXPLANS(2,4,3,12,3,3,2,6);
```

```
  TRDEUXPLANS(0,1,0,5,3,3,4,24);
  TRDEUXPLANS(0,0,1,2,1,0,0,3);
  TRDEUXPLANS(2,3,4,0,1,4,2,0);
  TRDEUXPLANS(0,1,2,0,2,3,2,0);
  TRDEUXPLANS(1,1,-1,2,2,3,3,8);
  TEXTMODE;AFFTEXT3(10);
  TRTROISPLANS(1,0,0,2,0,2,0,5,0,0,3,7);
  TRTROISPLANS(3,2,3,12,1,2,2,6,6,2,3,18);
  TEXTMODE;AFFTEXT3(11);
  TRTROISPLANS(0,2,3,4,0,2,3,8,1,0,2,6);
  TRTROISPLANS(0,1,0,1,0,2,3,12,0,0,1,1);
  TEXTMODE;AFFTEXT3(12);
  TRTROISPLANS(3,6,4,12,3,6,4,12,6,4,3,12);
  TRTROISPLANS(1,0,0,1,0,0,1,1,1,0,1,2);
  TRTROISPLANS(1,2,2,6,1,1,1,4,2,1,1,6);
  TEXTMODE;AFFTEXT3(13);
END.
```

```
(*$S+,N+*)
PROGRAM GRAPH;
USES APPLESTUF,UTILUNIT,TURTLEGRAPHICS,
     DESSIN22,
     DECLAR33,CHOIXPLAN,INTER33,DESSIN33;

VAR A1,B1,C1,D1,A2,B2,C2,D2,A3,B3,C3,D3:REAL;
    C:CHAR;
    CHOIX:INTEGER;
    ARRET:BOOLEAN;

PROCEDURE COEF3(A,B,C,D:REAL);FORWARD;

SEGMENT PROCEDURE QUELCHOIX;
BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,2);
  WRITELN('        PROGRAMME GRAPHIQUE : ');
  WRITELN('       ************************');
  WRITELN;
  WRITELN('      VOUS POUVEZ : ');
  WRITELN;
  WRITELN('1 REPRESENTER DES DROITES');
  WRITELN;
  WRITELN('2 REPRESENTER DES SYSTEMES 2*2');
  WRITELN;
  WRITELN('3 REPRESENTER DES PLANS');
  WRITELN;
  WRITELN('4 REPRESENTER DES PLANS SUPERPOSES');
  WRITELN;
  WRITELN('5 REPRESENTER DES SYSTEMES 3*3');
  WRITELN;
  WRITELN('6 QUITTER CE PROGRAMME');
  GOTOXY(5,20);
  WRITELN('QUEL EST VOTRE CHOIX (1-6) ?');
  READ(KEYBOARD,C);
  WHILE (ORD(C)<49) OR (ORD(C)>54)
  DO BEGIN
       GOTOXY(5,22);
       WRITELN('REPONSE ENTRE 1 ET 6 S.V.P.');
       READ(KEYBOARD,C)
     END;
  CHOIX:=ORD(C)-48
END;

SEGMENT PROCEDURE SAISIE(NROSAISIE:INTEGER);

   PROCEDURE SAISIE2;
   BEGIN
     CASE NROSAISIE OF
      1:BEGIN
          WRITE('A ?');READLN(A1);
          WRITE('B ?');READLN(B1);
          WRITE('C ?');READLN(C1);
        END;
      2:BEGIN
          WRITE('A1 ?');READLN(A1);
          WRITE('B1 ?');READLN(B1);
          WRITE('C1 ?');READLN(C1);
          WRITE('A2 ?');READLN(A2);
          WRITE('B2 ?');READLN(B2);
          WRITE('C2 ?');READLN(C2);
        END;
      3:BEGIN
```

```
            WRITE('A ?');READLN(A1);
            WRITE('B ?');READLN(B1);
            WRITE('C ?');READLN(C1);
            WRITE('D ?');READLN(D1);
          END;
        4:BEGIN
            WRITE('A1 ?');READLN(A1);
            WRITE('B1 ?');READLN(B1);
            WRITE('C1 ?');READLN(C1);
            WRITE('D1 ?');READLN(D1);
            WRITE('A2 ?');READLN(A2);
            WRITE('B2 ?');READLN(B2);
            WRITE('C2 ?');READLN(C2);
            WRITE('D2 ?');READLN(D2);
          END;
        5:BEGIN
            WRITE('A1 ?');READLN(A1);
            WRITE('B1 ?');READLN(B1);
            WRITE('C1 ?');READLN(C1);
            WRITE('D1 ?');READLN(D1);
            WRITE('A2 ?');READLN(A2);
            WRITE('B2 ?');READLN(B2);
            WRITE('C2 ?');READLN(C2);
            WRITE('D2 ?');READLN(D2);
            WRITE('A3 ?');READLN(A3);
            WRITE('B3 ?');READLN(B3);
            WRITE('C3 ?');READLN(C3);
            WRITE('D3 ?');READLN(D3);
          END;
      END;
    END;

BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('VEUILLEZ INTRODUIRE VOS COEFICIENTS:');
  SAISIE2;
END;


SEGMENT PROCEDURE TRTROISPLANS(A1,B1,C1,D1,
                               A2,B2,C2,D2,
                               A3,B3,C3,D3:REAL);
VAR M1:MATRICE;
    M2:MATREEL;

BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('PRESENTATION DU SYSTEME 3*3');
  COEF3(A1,B1,C1,D1);
  COEF3(A2,B2,C2,D2);
  COEF3(A3,B3,C3,D3);
  MODE:=1;
  M2[1,1]:=A1;
  M2[1,2]:=B1;
  M2[1,3]:=C1;
  M2[1,4]:=D1;
  M2[2,1]:=A2;
  M2[2,2]:=B2;
  M2[2,3]:=C2;
  M2[2,4]:=D2;
```

```
  M2[3,1]:=A3;
  M2[3,2]:=B3;
  M2[3,3]:=C3;
  M2[3,4]:=D3;
  SUITE;
  SYST33(M1,M2);
END;

SEGMENT PROCEDURE TRDEUXPLANS(A1,B1,C1,D1,
                        A2,B2,C2,D2:REAL);
BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('    SUPERPOSITION DES 2 PLANS');
  COEF3(A1,B1,C1,D1);
  COEF3(A2,B2,C2,D2);
  SUITE;
  AA[1]:=A1;
  BB[1]:=B1;
  CC[1]:=C1;
  DD[1]:=D1;
  AA[2]:=A2;
  BB[2]:=B2;
  CC[2]:=C2;
  DD[2]:=D2;
  MULX:=-2/3;
  MULY:=-2/3;
  DEUXPLANS(1,2,
            AA[1],BB[1],CC[1],DD[1],
            AA[2],BB[2],CC[2],DD[2]);
  SUITE;
END;


PROCEDURE TRUNPLAN(A,B,C,D:REAL);
BEGIN
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('    REPRESENTATION DE L''EQUATION :');
  COEF3(A,B,C,D);
  SUITE;
  UNPLAN(A,B,C,D,1);
  SUITE;
END;


PROCEDURE COEF3;
VAR PLUSMOINS:CHAR;
BEGIN
  WRITE('     ',ROUND(A),' X1 ');
  IF B<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:='+';
  WRITE(PLUSMOINS,ABS(ROUND(B)),' X2 ');
  IF C<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:='+';
  WRITE(PLUSMOINS,ABS(ROUND(C)),' X3 ');
  IF D<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:=' ';
  WRITELN('=',PLUSMOINS,ABS(ROUND(D)));
END;
```

```
PROCEDURE COEF2(A,B,C:REAL);
VAR PLUSMOINS:CHAR;
BEGIN
  WRITE('         ',ROUND(A),' X1 ');
  IF B<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:='+';
  WRITE(PLUSMOINS,ABS(ROUND(B)),' X2 ');
  IF C<0 THEN PLUSMOINS:='-'
         ELSE PLUSMOINS:=' ';
  WRITELN('=',PLUSMOINS,ABS(ROUND(C)));
END;

PROCEDURE UNEDROITE(A,B,C:REAL);
VAR FR:FRACTION;
BEGIN
  RECALC:=FALSE;
  NDR:=1;
  MODE:=0;
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('     PRESENTATION DE L''EQUATION :');
  COEF2(A,B,C);
  SUITE;
  GR22(FR,FR,FR,A,B,C);
  SUITE
END;

PROCEDURE DEUXDROITES(A1,B1,C1,A2,B2,C2:REAL);
VAR MF:MATRICE;
    MR:MATREEL;

BEGIN
  MODE:=1;
  MR[1,1]:=A1;
  MR[1,2]:=B1;
  MR[1,3]:=C1;
  MR[2,1]:=A2;
  MR[2,2]:=B2;
  MR[2,3]:=C2;
  PAGE(OUTPUT);
  TEXTMODE;
  GOTOXY(0,10);
  WRITELN('     PRESENTATION DU SYSTEME 2*2 :');
  COEF2(A1,B1,C1);
  COEF2(A1,B1,C1);
  SYST22(MF,MR);
  SUITE
END;



BEGIN
  ARRET:=FALSE;
  INITORI;
  WHILE NOT ARRET
   DO BEGIN
        QUELCHOIX;
        CASE CHOIX OF
         1:BEGIN
           SAISIE(1);
           UNEDROITE(A1,B1,C1);
           END;
```

```
        2:BEGIN
          SAISIE(2);
          DEUXDROITES(A1,B1,C1,A2,B2,C2);
          END;
        3:BEGIN
          SAISIE(3);
          TRUNPLAN(A1,B1,C1,D1);
          END;
        4:BEGIN
          SAISIE(4);
          TRDEUXPLANS(A1,B1,C1,D1,
                      A2,B2,C2,D2);
          END;
        5:BEGIN
          SAISIE(5);
          TRTROISPLANS(A1,B1,C1,D1,A2,B2,
                       C2,D2,A3,B3,C3,D3);
          END;
        6:ARRET:=TRUE;
      END;
    END
END.
```